PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-285154
(43)Date of publication of application : 23.10.1998

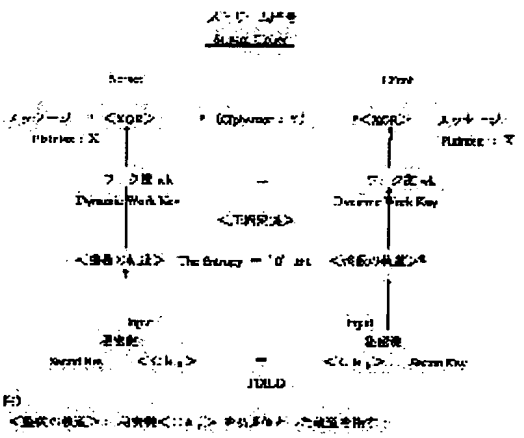
(51)Int.CI. H04L 9/26
G09C 1/00
G09C 1/00
H04L 9/08

(21)Application number : 09-090873
(22)Date of filing : 09.04.1997

(71)Applicant : METEOOLA SYST KK
AIRU:KK
(72)Inventor : WATANABE EIJI
SEKIGUCHI YOSHIHIRO

(54) KEY GENERATION SYSTEM CONSTITUTING PERFECT SECRECY CIPHER SYSTEM, KEY SHARED PROTOCOL WITH AUTHENTICATION, 'ONE-TIME STREAM CIPHER', 'ONE-TIME PASSWORD' AND KEY MANAGEMENT ALGORITHM

(57)Abstract:
PROBLEM TO BE SOLVED: To give informational logical security to an open network.
SOLUTION: In a key generation system which provides both terminals that perform communication in a client/server network, etc., with a register which operates with 2u (u>4) adic number, and is provided with a mapping algorithm part that executes map of an integer space of the register (Is → Is) to its own and generates an 'integer orbit' and a compressing and converting part which compresses and converts the 'integer orbit' and creates a binary 'key stream', when inverse map that traces back a map algorithm to the past is attempted, not only 'n-bits entropy' occurs, but also 'informational logical security map algorithm' to which 'entropy' that accompanies compressive conversion is also added is mounted to n-times inverse map and is 'intentionally' embedded in the algorithm of cipher and decoding.



LEGAL STATUS

[Date of request for examination] 09.04.2004

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（Ａ）**　(11)特許出願公開番号

特開平10－285154

(43)公開日　平成10年(1998)10月23日

(51)Int.Cl.[6]		識別記号	FI		
Ｈ０４Ｌ	9/26		Ｈ０４Ｌ	9/00	６５９
Ｇ０９Ｃ	1/00	６１０	Ｇ０９Ｃ	1/00	６１０Ｄ
		６５０			６５０Ｂ
Ｈ０４Ｌ	9/08		Ｈ０４Ｌ	9/00	６０１Ｅ

審査請求　未請求　請求項の数６　ＯＬ　（全 25 頁）

(21)出願番号　特願平9－90873

(22)出願日　平成９年(1997)４月９日

(71)出願人　591223231
メテオーラ・システム株式会社
神奈川県横浜市港北区高田町1549番地

(71)出願人　596059163
株式会社アイル
東京都中野区本町２丁目50番８号

(72)発明者　渡邊　榮治
神奈川県横浜市港北区高田町1549番地

(72)発明者　関口　義浩
神奈川県伊勢原市石田200　メテオーラ・システム株式会社内

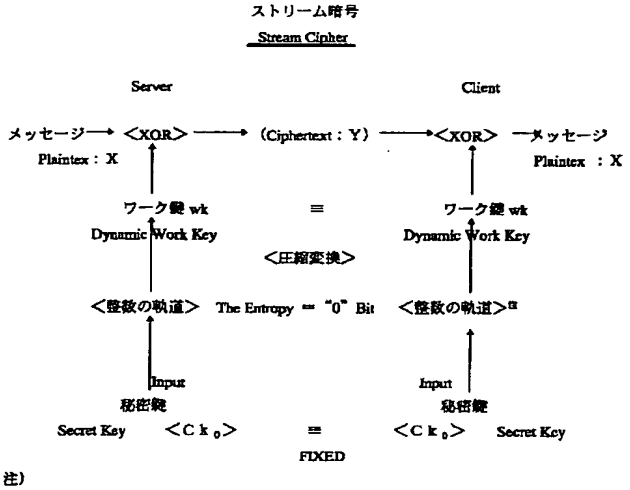
(74)代理人　弁理士　三好　秀和　（外８名）

(54)【発明の名称】　完全守秘性暗号系を構成する鍵生成システム、認証付き鍵共有プロトコル、“Ｏｎｅ－Ｔｉｍｅ Ｓｔｒｅａｍ Ｃｉｐｈｅｒ”、“Ｏｎｅ－Ｔｉｍｅ ｐａｓｓｗｏｒｅｄ”及び鍵管理アル

(57)【要約】　（修正有）

【課題】　オープンネットワークに情報理論的な安全性“Unconditional Security”を与える。

【解決手段】　クライアント／サーバー・ネットワーク等で通信を行う端末双方に、２ｕ（ｕ＞４）進数で演算するレジスターを備え、レジスターの整数空間Ｉｓのそれ自身への写像（Ｉｓ→Ｉｓ）を実行して「整数の軌道」を生成する写像アルゴリズム部、及び「整数の軌道」を圧縮変換して２進の「鍵ストリーム」を生成する圧縮変換部を備えた鍵生成システムにおいて、写像アルゴリズムを過去に遡る逆写像を試みると、ｎ回の逆写像に対しては“ｎビットのEntropy”が生じだけでなく、圧縮変換に伴う“Entropy”もそれに加わる「情報理論的に安全な写像アルゴリズム」が実装され、暗号と復号のアルゴリズムの中に「意図的に」埋め込む。



注)
<整数の軌道>：秘密鍵<Ck0>を初期値とした軌道を指す；

1

【特許請求の範囲】

【請求項１】　クライアント／サーバー・ネットワーク（以下Ｃ／Ｓネットワーク）などで通信を行う端末双方に、２ｕ（ｕ＞４）進数で演算するレジスターを備え、該レジスターの整数空間Ｉｓのそれ自身への写像（Ｉｓ→Ｉｓ）を実行して「整数の軌道」を生成する写像アルゴリズム部と、該２ｕ進数の「整数の軌道」を圧縮変換して２進の「鍵ストリーム」を生成する圧縮変換部、とを備えた鍵生成システムを実装し、

該鍵生成システムには、その写像アルゴリズムを過去に遡る逆写像を試みると、ｎ回の逆写像に対しては“ｎビットのEntropy”が生じだけでなく、上記圧縮変換に伴う“ Entropy”もそれに加わる「情報理論的に安全な写像アルゴリズム」が実装されており、

該情報理論的に安全な写像アルゴリズムを「鍵の情報理論的な安全性」を追求する目的で、暗号と復号のアルゴリズムの中に「意図的に」埋め込んだ事を特徴とする情報理論的に安全な対称暗号系。

【請求項２】　前記対称暗号系には、

該写像アルゴリズムによる整数軌道の任意の軌道値“ｃｋ0”を「秘密鍵」として共有すると同時に、実際に暗号／復号するワーク鍵には上記「鍵ストリーム」を使う一方、

秘密鍵＜ｃｋ0＞がどちらかの該鍵生成システムにインプットされると、該秘密鍵を情報理論的に安全な秘密鍵にするために、該写像アルゴリズムを規定回数（ｉ回）実行させて、該秘密鍵＜ｃｋ0＞の整数軌道（ｃｋ1、ｃｋ2、ｃｋ3、、、ｃｋｉ）を作らせると共に、該整数軌道値を該端末内に「保存」するという規約と、

さらに、この保存されているｉ番目の整数軌道値＜ｃｋｉ＞を初期値とした新たな整数軌道（ｃｋｉ、ｃｋｉ+1、、、ｃｋｉ+ｊ）を生成させて、この後者の整数軌道を圧縮変換した「鍵ストリーム」をワーク鍵に使うという規約と、

これら二つの規約から構成される「安全な同期プロトコル：Secure Synchronous　Protocol：ＳＳＰ」を実装し、

該「安全な同期プロトコル」を共有する事により、対称暗号系の共通鍵の安全性を計算量の安全性“Computational Security”ではなく、情報理論的に安全な、つまり“Unconditional Security”の問題に帰着させたたことを特徴とする対称暗号系。

【請求項３】　秘密鍵＜ｃｋ0＞を共有している前記

2

１、２項記載の対称暗号系は、互いに相手を確認しうる手段（ＩＤ）として共有している秘密鍵＜ｃｋ0＞を、認証の手段としてネットワーク上で交換せず、保持すると共に、

対称暗号系のサーバー上で他の偶然的イベントを初期値とした別の整数の軌道値を生成し、これを前記安全な同期プロトコル（ＳＳＰ）の下でサーバーが保持しているクライアントの秘密鍵＜ｃｋ0.serv＞によって暗号化し、該秘密鍵＜ｃｋ0.serv＞の「代理の鍵＜ｓｋｉ.serv＞」として他方のクライアントに配送する一方、

この暗号データを受け取ったクライアントも、該クライアントが保持していた秘密鍵＜ｃｋ0.clin＞により同じく前記“ＳＳＰ”の下で該暗号データを復号化して「代理鍵」＜ｓｋｉ.clin＞を受け取るという規約「鍵の同期配送プロトコル」を実装し、

さらに、サーバーは代理鍵＜ｓｋｎ.serv＞を初期値とした整数軌道を生成して該拡張代理鍵群を保持し；

【数１】｛ｓｋ（ｎ＋１）.serv、ｓｋ（ｎ＋２）.serv、ｓｋ（ｎ＋３）.serv、、｝、

一方クライアントも代理鍵＜ｓｋｎ.clin＞について、それを初期値とした 整数軌道を生成して該拡張代理鍵群を保持し；

【数２】｛ｓｋ（ｎ＋１）.clin、ｓｋ（ｎ＋２）.clin、ｓｋ（ｎ＋３）.clin、、｝　、

これら拡張代理鍵群をそれぞれ前記鍵生成システムの圧縮変換部に入力し、それぞれ対応する拡張ワーク鍵系列を生成し、かつ保持し；

【数３】｛、、ｗｋ１.serv、ｗｋ２.serv、ｗｋ３.serv、、、｝

｛、、ｗｋ１.clin、ｗｋ２.clin、ｗｋ３.clin、、、｝

これら情報理論的に安全な拡張ワーク鍵群をネットワークへ出して交換するという規約「鍵の同期認証プロトコル」を実装し、

該「鍵の同期配送プロトコル」と「鍵の同期認証プロトコル」の両者から構成される「認証付きの鍵共有プロトコル：Authenticated Key Agreement Protocol」を実装する事を特徴として秘密鍵の共有を実現する前記１、２項記載の対称暗号系。

【請求項４】　対称暗号系の端末双方が“login”後、前記１、２項記載の安全な同期プロトコル（ＳＳＰ）の下で前記３項記載の「認証付き鍵共有」を成した後に於いては、当該対称暗号系の中の鍵群は、

拡張ワーク鍵の一致 ＝ 拡張秘密鍵を共有する、

【数４】

＝ ＜ｃｋｉ.serv＞ ＝ ＜ｃｋｉ.clin＞ ； ｉ＝1～任意

＝ ＜ｃｋ0.serv＞ ＝ ＜ｃｋ0.clin＞；

＝ ＜ｃｋ0＞；秘密鍵を共有する、

を無条件に満足する共通鍵群となるので、

“login”が発生する度に、端末双方とも一回使用した秘密鍵を該共通鍵群の中の未使用の拡張秘密鍵をもって「更新する」ことを特徴とした対称暗号系又は、前記

１、２、３項記載の構成要件を満たす“One-Time Stream Cipher”、同じく前記１、２、３項記載の構成要件を満たすことを特徴とした“One-Time Password”。

【請求項５】　多数のクライアントの秘密鍵のファイル「鍵箱」を備えたサーバーが、その鍵箱を情報理論的に安全なセキュリテイで守るために、前記２項記載の安全な同期プロトコル（ＳＳＰ）の下で前記３項の「認証付き鍵共有」を成した後に於いて、

該サーバーは、「鍵箱」の任意の秘密鍵＜ｃｋ0＞を前記１項記載の当該秘密鍵＜ｃｋ0＞の「鍵ストリーム」＜ｗｋ-ｃｋ0＞によって該鍵＜ｃｋ0＞自身を暗号化し、該「鍵箱」を暗号ファイルCRYPTO（ｃｋｎ：ｎ＝0,1,2,3,,）として管理する一方、

前記１乃至４項記載の対称暗号系がこれを復号化するには、該クライアント自身が保持している鍵＜ｃｋ0.clin＞から生成した前記１、２項記載の鍵ストリーム＜ｗｋ-ｃｋ0.clin＞しか復号化の手段が残されていないので、

前記１乃至４項記載の対称暗号系は、次の“login”時に、該クライアントの該鍵ストリームをサーバーに配送し、

該サーバー上で暗号ファイル CRYPTO（ｃｋｎ：ｎ＝0）を復号化し、当該クライアントの秘密鍵＜ｃｋ0.serv＞をサーバーへ与えるとした「鍵管理アルゴリズム」を備え、

これにより該サーバーは該秘密鍵＜ｃｋ0.serv＞を入手した後には、前記３項記載の規約「認証付き鍵共有プロトコル」の実行に入ることを特徴とした鍵管理アルゴリズム及び鍵サーバー。

【請求項６】　前記１項記載の鍵生成システムのレジスターは、10^n（ｎ＞＝１）個または2^n個（ｎ＞１）の桁から成立する一様レジスターであり、その一桁も又10^m個または2^m個の桁から成立する（10^n＜＝10^m）一様レジスターであり、というように「入れ籠構造」を成した「自己相似形レジスター」であることを特徴とする一方、

該鍵生成システムの写像アルゴリズムは、該一様レジスターを10^n進数または2^n進数の整数空間Ｉｓと見なす該整数空間のそれ自身への写像演算Ｆである；（Ｆ：Ｉｓ→Ｉｓ）。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、従来のネットワーク・セキュリティーが計算量的安全性“Computational Security”に依存していたのに対して、オープンネットワークに情報理論的な安全性“Unconditional Security”を与える実装技術に関する。

【０００２】①　情報理論的な安全な鍵生成システム
②　情報理論的な安全な認証技術
③　情報理論的な安全な鍵配送技術

④　情報理論的な安全な鍵保管技術
⑤　情報理論的な安全なメッセージの暗号技術
⑥　情報理論的な安全なデジタル署名技術
に亘る。

【０００３】

【従来の技術】インターネットにしろイントラネットにしろ、そこで必要とされるセキュリティは、上述６つの分野に亘る。これらセキュリティーを実現する暗号技術には対称暗号方式と非対称暗号方式とがあり、現在この二方式が合い補ったハイブリッド方式がネットワーク・セキュリティーを達成したかに見えている。対称暗号方式は、暗号鍵と復号鍵とが同じ対称鍵である。メッセージの機密保持を確保するためには、この対称鍵を秘密にしなければならないので、これを「秘密鍵」と呼んでいる。

【０００４】歴史上ネットワーク上で、この秘密鍵をどのようにして共有するかを解決したのが、Diffe-Hellmanの鍵配送方式である。又、ここからＲＳＡも工夫され、非対称暗号そのものが確立したことはよく知られている。

【０００５】現在、上述６つの技術分野は、⑤のメッセージの暗号に対称暗号を適用し、その他の分野は非対称暗号の応用技術を適用するというハイブリッド方式によってカバーされている。この内、②の個人認証技術に関しては、実用上もっと手軽な技術が普及しつつある。

【０００６】例えば、“challenge & Response”という手法を適用して、クライアントがサーバーにアクセスすると、毎回サーバーは“challenge number”という平文を送り出し、これを受け取ったクライアントは“password”という暗号文をサーバーに返す。これがResponseである。クライアントとサーバー双方にはＤＥＳなどの対称暗号ソフトがあらかじめインストールされていることは言うまでもない。

【０００７】毎回平文と暗号文とがネットワーク上に送出されているので、このセキュリティーを破ることは国家機関にとっては容易である。一方、民間ではこれで十分のセキュリティーを達成しているようである。もっと信頼の於ける認証技術としては、One-Time password (OTP; RFCにてOTPと称されている）がある。難点は、使用回数が限られているので、商取引には向かない。ＲＳＡが現在主流の認証技術である。しかし、これは計算量的安全性に依存していることを忘れてはならない。

【０００８】

【発明が解決しようとする課題】従来知られている暗号は“計算量的に安全である”と主張している技術である；これは、「Computational Security」と言われている。この種の技術が、現在オープンネットワークのスタンダードになっている。例えば、DESやRSAが有名である。

【０００９】それはルーチンワークの大好きなコンピュ

ータで破ろうとする限り、安全であることが保証されている暗号である。しかし、RSAに関して言えば、将来素数分布に関する新しい数学が生まれたらどうなるであろうか（リーマン予想）。整数の素因数分解を容易に行える様になるであろうし、その時、世界中を飛び回っている電子マネーは、電子マネーでなくなる日となる。“計算量的に安全である“とは、この様な意味での相対的安全性であって、将来に亘って安全であるという意味ではない。

【００１０】もう一つ困ったことは、従来の暗号技術は国家の安全保証とリンクしていることである。従って、これら暗号技術の民間ユースの転用に対して色々と制限が入ってくるのは当然である。

【００１１】故に、オープンネットワークで使える新しい暗号技術の出現が望まれる。この世界でスタンダードになるセキュリティを実現できる暗号技術は、どんな条件に耐えねばならないであろうか；その技術が全ての人に公開され、評価され、誰でもそのソフトウェアが入手可能になるという事態を、まず受入なければならない。

【００１２】かってのように一部の人達だけが技術を共有し、つまり技術そのものを非公開の扱いにすることに依ってセキュリティーを保つという秘密主義は、オープンネットワークに通用しない。従って、単に暗号システムを複雑にすることによっては、問題の解決にならないことが判る。

【００１３】Gilbert Vernumによって１９１７年に発表されたというOne-Time Padは、単純にして、かつ“完全守秘性”注[1]を提供する最初の実用的な技術であることが知られている。しかし、この技術は上述のスタンダードに耐えうる条件を備えているにもかかわらず、オープンネットワークに適用できない深刻な問題を内包している。そのアポリアとは、「鍵の配送と鍵の管理」に関することである；これは、鍵を配送するのに当該通信路以外に、別に安全な通信路を必要とするというアポリアである。

【００１４】従って、非公開の扱いを可能とし、鍵の配送を特別な人が行う軍事、外交上の通信のみに使われているということである。完全守秘性の概念は、この暗号の実用化に後れて３０年後に、C.E.Shannon（注1）によって与えられたという。この意義は、今日オープンネットワークの時代に特に大きい。何故なら、従来のオープンネットワークに於ける暗号技術が「Computational Security」という相対的 安全性に依存しているのに対して、「確率論的な答え」しか許さない暗号系の理論的な存在とその安全性を提示したことである。判りやすく言えば、その暗号技術を全て公開し、裏も表もなく公開し、あらゆる技術進歩を計算に入れても、その暗号を“破れる”のは、唯一確率論しか無いと言う暗号の安全性を明らかにしたことである。このような暗号系は「情報理論的に安全である」とか、「Unconditional Security」であると言われている。俗な言い方をすれば、ハッカーは、サイコロの目が2^{256}もあるようなサイコロを振る予想しか出せないし、コンピ ュータのプログラムは暗号文を解析して平文を作ったとしても、あらゆる平文が等確率に現れるから、やはりサイコロを振る結果になる。

【００１５】結論を言えば、オープンネットワークでスタンダードになれるセキュリティー技術とは、こういう“Unconditionally Secure”な暗号系に依存すべきである。

【００１６】しかし、この種の安全な暗号系、例えば“ One-Time Pad cipher”をネットワークへ適用しようとすると（注3）、「鍵の配送と鍵の管理」を他の暗号系または他 の安全な通信路に依存しなければならないという深刻な問題に出会うので、このアポリアをどのように始末するか、ここが新しい暗号理論とその実装手段の試金石になるところであるし、本発明がブレイクスルーした点でもある。

【００１７】

【課題を解決するための手段】暗号技術の基本は鍵生成システムにある、言い換えると、暗号技術の基本は乱数生成のアルゴリズムに依存する。従来の乱数生成アルゴリズムは“Computationally Secure”であっても、決して“Unconditionally Secure”な乱数生成法ではない、あるいは情報理論的な安全性の側面から研究された痕跡がない。

【００１８】この“Unconditionally Secure”な乱数生成理論の一つが、「カオス発生の原因であるフラクタル構造そのもの」の中に発見された。カントール集合は「情報理論的に安全な」なアルゴリズムであるという発見である。単に、カオス的現象作り出してそれから乱数列を取り出すという意味ではない。言い換えれば、コンピュータでカオス的乱数を作り、乱数列を生成したと言う人が居ても、それが「情報理論的に安全な」乱数生成手段であるという主張にはならないし、また情報理論的に安全な乱数の利用をしているという主張にもならない。さらにカオス理論で作った乱数なら安全であるとか、逆に簡単な非線型関数を使っているから安全ではないなどの議論も、この“Unconditional Security”関する議論と全く関係ない。この点は特に強調しておかねばならない。

【００１９】１）カオス写像の情報理論的に安全な一方向変換性；どのような非線形曲線を使うにしろ、カントール集合を成す数学上のカオス写像によって生成されたカオス軌道は（Chaos Orbit）、未来に向かっては一意に決定される一方、過去に向かうと（ｎ回逆写像しようと試みると）2^nの不確定度を生じる（カントール集合を逆にたどる）。このｎ回逆写像の“Entropy”を考えてみよう。

【００２０】カオス軌道は互いに交わらない、つまり同

じ数学上の点を通らないという性質があるので、有限の時間内で「カオス軌道上の点は全て等確率で出現する」と言える。このカオス写像の性質を情報理論の視点で見ると、「ｎ回逆写像のEntropyはｎビットである」ことが判る（等確率で出現するから）。ここで注意すべきは、数学上のカオス軌道と実装技術上のカオス軌道とは一致しない；何故なら実装技術上のカオス軌道は有限精度の整数の軌道であるから、単にカオス理論をプログラムしてもカオスにならないのである。そこで話を数学上のカオスに限るなら「過去のカオス軌道値は、現在の軌道値よりｎビット余計な不確定度を帯びる、つまり、ｎビットの情報量を担う」という情報理論的な構造を発見する。（この情報理論的な構造は必ずしもカオス写像のみに発見されるとは限らないが、カオス理論はすでに確立しているので安心して使える。故に、以下カオス写像を例にして“Unconditionally Secure”な鍵生成理論を解説する。）

この情報理論的な構造に著目すると、我々は任意の瞬間のカオス軌道値そのものが「情報理論的に安全な秘密鍵」として使えるというアイデアを発見する。何故なら暗号を破るという行為は、“Recipient”側にある現在のデーターから、「過去に遡って」“Originator”側の原因を特定する作業であるから、ｎ回逆写像の“Entropy”がｎビットであるとは、「情報理論的に安全な鍵」を提供するアルゴリズムになる。一方この秘密鍵は、そのカオス軌道を「未来に向かっては一意に決定する」から「復号鍵： unique—decipherbility 」として機能させる事も出来る。これは暗号学的には、真に出来すぎた属性である。ここに改めて、我々は「復号鍵」として機能するカオス軌道上の任意の瞬間のカオス軌道値を「情報理論的に安全な拡張秘密鍵：Secret Key-Enhansed」と名付けると共に、該拡張秘密鍵から見て、ｎビット余計な不確定度を帯びている過去のカオス軌道値を「秘密鍵」と定義する。これらのカオス軌道値を、アイデア上の「カオス鍵」と言う場合がある。（詳細は図—２にて説明する）。上記のように、数学上のカオス写像をC.E. Shannon.の情報理論注１から再検討すると、カオス軌道は「情報理論的に安全な一方向変換性を備えている」ことが判る。この情報理論的発見から、我々は完全守秘性暗号系“Perfect SecrecyCryptosystem”に関する基礎理論を確立した。このカオス写像自体に関しては、既に思想として、１９９0年特願平２—２２６５００「無線式電話の秘話システム及び秘話通信用LSI」の中で“秘話ＯＳ”として提示し説明しているところである。

【００２１】今回、視点を.Shannoの研究に移し、カントール集合を成すアルゴリズムを情報理論的な安全性を実現する基礎理論として採用した。

【００２２】実装技術上のカオス写像を「カオス的写像」注[2]と表現する。単にカオス写像という時は数学上の「情報理論的に安全な一方向変換性」を意味すると同時に、そのカオス写像を有限精度のコンピュータに実装する場合「カオス的写像」と表現する。従って、「カオス的写像」とは、それが未来に向かう時その情報量は“零ビット”であり、過去に向かう時は“ｎビット”の情報量を必要とすることが「期待出来る」実装技術上のアルゴリズムである。該実装技術上のアルゴリズムは必ずしもカオス的写像であるとは限らないが、カオス的写像アルゴリズムから生み出される鍵を「カオス的鍵；chaotic key」と言う場合がある。

【００２３】実施例で示すように、該実装技術では、カオス的写像の未来に向かうその情報量が“零ビット”であることを、暗号／復号を「正確に実行する」根拠とし、認証を「代理鍵」によって「正確に実行する」根拠とし、従って代理鍵を「共有する」根拠とし、また秘密鍵を「一回しか使えない」ように「正確に同期して更新する」根拠とする一方、またカオス的写像が過去に向かうと“ｎビット”の情報を必要とする事を、「情報理論的に安全な秘密鍵」を生み出す根拠にしている。これら全てカオス的鍵の性質に依存する。

【００２４】“One-Time Pad Cipher”をネットワークへ適用する際に生じる深刻な鍵管理の問題は、全て情報理論的に安全な秘密鍵の性質を適用して解決した。すなわち、未来に向かう写像アルゴリズムは一意に拡張秘密鍵を決めるという性質を適用して、ＴＣＰ／ＩＰなどの一般的なプロトコル上に鍵生成システムを実装して「情報理論的に安全な同期制御」を実施しブレイクスルーしたものである。ここで、同期制御とは、

① 鍵ストリーム（keystream）が平文と独立に選ばれることと、

② クライアント・サーバー双方で実行されるカオス的写像をプロトコルが同期して制御する、という二つの意味を代表する。

【００２５】この通信プロトコル上の同期制御を「安全な同期プロトコル：Secure Synchronous Protocol：ＳＳＰ」というが、これによって「認証付きの鍵共有：Authenticated Key Agreement」がオープン・ネットワーク上で可能になる。このユニークな鍵共有メカニズムによれば「鍵を交換するためには、別に安全な通信路を設けなければならない」という“One-Time Pad Cipher”に特有な鍵配送問題を解決できるのである。

【００２６】２）鍵生成システム；有限精度のパソコンで鍵を等確率に作れるか；カントール集合を生み出す母胎は「連続量」である。ロジスティック曲線のように単純な曲線であるからという理由で、その鍵生成を軽く見る暗号専門家も居るかも知れないが、それはカオス生成の根底にある「連続量とフラクタルな構造」を見落としているからである。逆に、このカントール集合を生み出す構造に注意を払わないで、単にカオス写像をプログラムしたとしても、カオスを作れないことは明らかであろう；連続量とコンピュータの有限精度との差は、宇宙の

大きさととゴミを比較する以上の差がある。従って、有限精度のパソコンで鍵を等確率に作れない。

【００２７】しかし、我々は従来のフラクタル構造にプラスして、「加算個の無限を取り扱うフラクタルな演算構造」（注2）をコンピュータの世界に導入することを考案した 。

【００２８】このフラクタルな演算理論に基づくアルゴリズムは多数考えられるが、どのようなアルゴリズムにしろ「鍵が等確率に出現する」根拠を持った工夫・考案が無ければならない。（特願平８―１０８０５８「周期性を考慮したカオス的乱数列の発生装置」及び特願平８―１６９８６９「乱数生成装置及び乱数生成システム並びに暗号通信方式」（注2）の乱数生成法は、“Computationally Secureな長期周 期軌道”に関する一方、前特願平８―３３８５８３「乱数列の発生方法」の乱数生成法は、非周期軌道に関する。

【００２９】該アルゴリズムは、いずれもプログラム技術としては多重演算アルゴリズムを利用しているが、それが“Chaotic Orbit”を生成する理論上の根拠に依存している。すなわち「加算個の無限を取り扱うフラクタルな演算構造」を継承してそれに準じている）。

【００３０】我々は「鍵が等確率に出現する」ことを期待できる他の工夫も歓迎する。以上要約すると、“Unconditionally Secure”な鍵生成手段とは、秘密鍵とワーク鍵とを生成する手段を備え、かつそれぞれ等確率に生成すること、及びそのプロセスが情報理論的に安全な一方向変換性を備えていることである。鍵が等確率に出現する状況証拠には、①“ keystream ”に自己相関が無い、②０／１ビットの偏りが無い（真正乱数）、演算レジスターに“Lorentz plot”が現れる（実施例参照）などがある。これらは「Perfect Secrecy」を提供する上で考慮されねばならない重要な要因である。

【００３１】次の４項でも触れる。

３）鍵ストリーム；動的ワーク鍵；上記鍵生成システムでは、整数の軌道値を通常１０進数で表現する一方、実際暗号に使うワーク鍵には、整数の軌道を圧縮変換した２進のビット列を使う。

【００３２】秘密鍵を１０進表現で、動的ワーク鍵を２進表現で使うと、この２進ビット列（子）からカオス的軌道（親）を特定しようとしても、言い換えればカオス的写像の内部情報を得たいと盗聴者が意図しても、圧縮変換に伴う２２ｎ の不確定度と、カオス的写像を過去に遡る2^{i} の不確定度が、共に乗算で効いてくるので、その情報を計算しても得られない、さらに情報理論的に安全になる；その不確定度は「2^{2n} ＊2^{i}」である。

【００３３】このような情報理論的に安全なワーク鍵を「動的ワーク鍵：Dynamic Work-key」または“secure keystream”と参照する。

【００３４】当該鍵生成システムの秘密鍵のEntropyは「２ｎ＋ｉビット」である。この不確定度は、丁度2^{2n+i} 個の“目”を持ったサイコロを振るに等しい。一例として、ｎ＝６４、ｉ＝１２８と仮定すると、そのサイコロの目は2^{256} 個ある。該秘密鍵を突き止めるのは、この巨大なサイコロを振るに等しい。

【００３５】４）“One-Time Stream Cipher”；対称暗号であるところのストリーム暗号に、上述の情報理論的に安全な鍵生成システムを組み込んで、あらゆる階層の鍵を一回使ったら捨てるとしたルール、つまりプロトコルを導入する。この暗号系を仮に“One-Time Stream Cipher”と言うことにする。

【００３６】一般に、ストリーム暗号では、ワーク鍵を平文と同じ長さのビットストリームとして生成すると共に、このワーク鍵と平文とを排他的論理和（XOR）し暗号文にする。

【００３７】今“One-Time Stream Cipher”の一つつの端末に著目して鍵の解析をして見れば、どの秘密鍵も動的ワーク鍵も、全ての鍵が一度使ったら捨てられる、“One-Time Use”に限られているはずである。この時、当該通信プロトコルに組み込まれている、上記鍵生成システムは、１）でも述べたように秘密鍵を全て等確率で算出し、当該通信プロトコルもそれを等確率で使う（使ったら捨てる）；とするなら、該暗号系は「Perfect Secrecy」を実現していることが期待出来る（注3）。

【００３８】実際、本発明の“One-Time Stream Cipher”では、生成した秘密鍵を“One-Time use”に限る一方、安定した通信も実現している。この安定した通信は、１）項で紹介したように、端末双方が情報理論的に安全な秘密鍵を用いて「安全な同期プロトコル：Secure Synchro―Protocol」を共有し、秘密鍵を拡張秘密鍵で更新するというダイナミズムを実現しているからである。以上１）～４）項は、「Perfect Secrecy 」を実現する暗号系の基礎理論の概要である。

【００３９】本特願はこの実装技術に関する：その構成は、

【表１】

【表１】

「ｐｅｒｆｅｃｔ　ｓｅｃｒｅｃｙ」の実装手段＝＝安全プロトコル；ｓｅｃｕｒｅ　ｓｙｎｃｈｒｏ－ｐｒｏｔｏｃｏｌ

- →情報理論的に安全な拡張秘密鍵：Secret Key-Extended
- →情報理論的に安全な鍵ストリーム：Secure Key Stream
- →認証付きの鍵共有プロトコル：Authenticated Key Agreement Protocol
  - →“One-Time Strean Cipher”
  - →“One-Time Password”
- →情報理論的に安全な鍵の管理アリゴリズム
- →カオス的鍵生成システム
  - ａ）全ての鍵が等確率で出現する
  - ｃ）動的ワーク鍵に偏りが無い：0.01％以下
  - ｄ）動的ワーク鍵に自己相関が無い：0.01％以下

【００４０】その実現手段は以下の通りである；

１．クライアント／サーバー・ネットワーク（以下Ｃ／Ｓネットワーク）などで通信を行う端末双方に、２ｕ（ｕ＞４）進数で演算するレジスターを備え、該レジスターの整数空間Ｉｓのそれ自身への写像（Ｉｓ→Ｉｓ）を実行して「整数の軌道」を生成する写像アルゴリズム部と、該２ｕ進数の「整数の軌道」を圧縮変換して２進の「鍵ストリーム」を生成する圧縮変換部、とを備えた鍵生成システムを実装し、

【００４１】該鍵生成システムには、その写像アルゴリズムを過去に遡る逆写像を試みると、ｎ回の逆写像に対しては“ｎビットのEntropy”が生じだけでなく、上記圧縮変換に伴う“ Entropy”もそれに加わる「情報理論的に安全な写像アルゴリズム」が実装されており、該情報理論的に安全な写像アルゴリズムを、「鍵の情報理論的な安全性」を追求する目的で、暗号と復号のアルゴリズムの中に「意図的に」埋め込んだ事を特徴とする情報理論的に安全な対称暗号系。

【００４２】２．記対称暗号系は、該像アルゴリズムによる整数の軌道の任意の軌道値“ｃｋ0”を「秘密鍵」として共有すると同時に、実際に暗号／復号するワーク鍵には上記「鍵ストリーム」を使う一方、秘密鍵＜ｃｋ0＞がどちらかの該鍵生成システムにインプットされると、該秘密鍵を情報理論的に安全な秘密鍵にするために、写像アルゴリズムを規定回数（ｉ回）実行させて、該秘密鍵＜ｃｋ0＞のカオス的軌道（ｃｋ1、ｃｋ2、ｃｋ3、、、、ｃｋｉ）を作らせると共に、該カオス的軌道値を該端末内に「保存」するという規約と、

【００４３】さらに、この保存されているｉ番目のカオス的軌道値＜ｃｋｉ＞すなわち「拡張秘密鍵」を初期値とした新たなカオス的軌道（ｃｋｉ、ｃｋｉ+1、、、ｃｋｉ+ｊ）を生成させて、この後者の整数軌道を圧縮変換した「鍵ストリーム」をワーク鍵に使うという規約と、これら二つの規約から構成される「安全な同期プロトコル：Secure Synchronous Protocol：ＳＳＰ」を実装し、

【００４４】該「安全な同期プロトコル」を端末群が共有する事により、対称暗号系の共通鍵の安全性が計算量の安全性“Computational Security”ではなく、情報理論的に安全性“Unconditional Security”の問題に帰着

させたことを特徴とする対称暗号系。

【００４５】３．秘密鍵＜ｃｋ0＞を共有している前記１、２項記載の対称暗号系は、互いに相手を確認しうる手段（ＩＤ）として共有している秘密鍵＜ｃｋ0＞を、認証の手段としてネットワーク上で交換せず、保持すると共に、対称暗号系のサーバー上で他の偶然的イベントを初期値とした別の整数の軌道値を生成し、これを前記1項の安全な同期プロトコル（ＳＳＰ）の下でサーバーが保持しいるクライアントの秘密鍵＜ｃｋ0.serv＞によって暗号化し、該秘密鍵＜ｃｋ0.serv＞の「代理の鍵＜ｓｋｉ.serv＞」として、他方のクライアントに配送する一方、

【００４６】この暗号データを受け取ったクライアントも、該クライアントが保持していた秘密鍵＜ｃｋ0.clin＞により同じく“SSP”の下で該暗号データを復号化して「代理鍵」＜ｓｋｉ.clin＞を受け取るという規約「鍵の同期配送プロトコル」を実装し、

【００４７】さらに、サーバーは代理鍵＜ｓｋｎ.serv＞を初期値としたカオス的軌道を生成して該拡張代理鍵群を保持し；

【００４８】

【数5】｛ｓｋ（ｎ＋１）.serv、ｓｋ（ｎ＋２）.serv、ｓｋ（ｎ＋３）.serv、、、｝、

【００４９】一方クライアントも代理鍵＜ｓｋｎ.clin＞について、それを初期値とした整数の軌道を生成して該拡張代理鍵群を保持し；

【００５０】

【数6】｛ｓｋ（ｎ＋１）.clin、ｓｋ（ｎ＋２）.clin、ｓｋ（ｎ＋３）.clin、、、｝、

【００５１】これら拡張代理鍵群をそれぞれ前記鍵生成システムの圧縮変換部に入力し、それぞれ対応する拡張ワーク鍵系列を生成し、かつ保持し；

【００５２】

【数7】｛、、ｗｋ１.serv、ｗｋ２.serv、ｗｋ３.serv、、、｝

｛、、ｗｋ１.clin、ｗｋ２.clin、ｗｋ３.clin、、、｝

【００５３】これら情報理論的に安全な拡張ワーク鍵群をネットワークへ出して交換する規約「鍵の同期認証プロトコル」を実装し；すなわち、

（１）クライアントからサーバーへ最も新しい「拡張ワーク鍵」＜ｗｋ３.clin＞を配送する。

【００５４】（２）サーバー上では下記の比較をする。

【００５５】クライアントのワーク鍵＜ｗｋ３.clin＞：サーバー上のワーク鍵＜ｗｋ３.serv＞

（３）一方サーバからもクライアントへ比較的に新しい「拡張ワーク鍵」を配送する。この鍵には＜ｗｋ１.serv＞又は＜ｗｋ２.serv＞どちらを使ってもよい。

（４）クライアント上では（３）項の鍵を使って下記の比較をする。

【００５６】サーバーのワーク鍵＜ｗｋ１.serv＞：クライアント上のワーク鍵＜ｗｋ１.clin＞

（５）一致を見なかった場合は、該端末の方のイニシャティブで通信を切断する。

【００５７】（６）一致を見た場合、

【００５８】

【数8】

対応する拡張ワーク鍵の一致 ： ＜ｗｋ３.serv＞ ＝ ＜ｗｋ３.clin＞
＝ ＜ｓｋn.serv＞ ＝ ＜ｓｋｎ.clin＞：代理鍵を共有し
＝ ＜ｃｋｉ.serv＞ ＝ ＜ｃｋi.clin＞ ：拡張秘密鍵を共有し
＝ ＜ｃｋ0.serv＞ ＝ ＜ｃｋ0.clin＞ ：秘密鍵を共有し
＝ 認証完了

【００５９】

などが成立する「鍵の同期認証プロトコル」を備え、
同時に
＝ ＜ｓｋｎ＞；を共有する、
すなわち、
＝ 代理鍵の情報理論的に安全な配送を完了する

とした「鍵の同期配送プロトコル」を実装し、該「鍵の同期配送プロトコル」と「鍵の同期認証プロトコル」の両者から構成される「認証付きの鍵共有プロトコル：Authenticated Key Agreement Protocol」を実装する事を特徴として秘密鍵の共有を実現する前記１、２項記載の対称暗号系。

【００６０】４．対称暗号系の端末双方が“login ”後、前2項記載の安全な同期プロトコル（ＳＳＰ）の下で前３項記載の「認証付き鍵共有」を成した後に於いては、当該対称暗号系の中の鍵群は、

【００６１】

【数9】

拡張ワーク鍵の一致 ＝ 拡張秘密鍵を共有し
＝ ＜ｃｋｉ.serv＞ ＝ ＜ｃｋi.clin＞；ｉ＝1~任意
＝ ＜ｃｋ0.serv＞ ＝ ＜ｃｋ0.clin＞；
＝ ＜ｃｋ0＞；秘密鍵を共有する、

【００６２】を無条件に満足する鍵群となるので、“login”が発生する度に、端末双方とも一回使用した秘密鍵を、該鍵群の中の未使用の任意の拡張秘密鍵もって「更新する」ことを特徴とした対称暗号系又は、前記１，２，３項記載の構成要件を満たす“One-Time Stream Cipher”同じく前記１，２，３項記載の構成要件を満たすことを特徴とした“One-Time Password”。

【００６３】５．多数のクライアントの秘密鍵のファイル「鍵箱」を備えたサーバーが、その鍵箱を情報理論的に安全なセキュリテイで守るために、前2項記載の安全な同期プロトコル（ＳＳＰ）の下で前3項の「認証付き鍵共有」を成した後に於いて、該鍵サーバーは、「鍵箱」の任意の秘密鍵＜ｃｋ0＞を前2項記載の当該秘密鍵＜ｃｋ0＞の「鍵ストリーム」＜ｗｋ-ｃｋ0＞によって該鍵＜ｃｋ0＞自身を暗号化し、

【００６４】該「鍵箱」を暗号ファイルCRYPTO（ｃｋｎ：ｎ＝0,1,2,3,,）として管理する一方、前記１乃至4項記載の対称暗号系がこれを復号化するには、該クライアントが保持している鍵＜ｃｋ0.clin＞から生成した上記鍵ストリーム＜ｗｋ-ｃｋ0.clin＞しか復号化の手段が残されていないので、前記１乃至4項記載の対称暗号系は次の“login”時に、該クライアントの該鍵ストリームをサーバーに配送し、該サーバー上で暗号ファイルCRYPTO（ｃｋｎ：ｎ＝0）を復号化し、当該クライアントの秘密鍵＜ｃｋ0.serv＞をサーバーへ与えるとした「鍵管理プロトコル」を実装する一方、該対称暗号系は、該サーバーが該秘密鍵＜ｃｋ0.serv＞を入手した後には、前3項記載の規約「認証付き鍵共有プロトコル」の実行に入ることを特徴とした鍵管理アルゴリズム及び鍵サーバー。

【００６５】６．前記１項記載の鍵生成システムのレジスターは、１０^{n}（ｎ＞＝１）個または2^{n}個（ｎ＞１）の桁から成立する一様レジスターであり、その一桁も又１０^{m}個または2^{m}個の桁から成立する（１０^{n}＜＝１０^{m}）一様レジスターであり、というように「入れ籠構造」を成した「自己相似形レジスター」であることを特徴とする一方、該鍵生成システムの写像アルゴリズムは、該一様レジスターを１０^{n}進数または2^{n}進数の整数空間Ｉｓと見なす該整数空間のそれ自身への写像Ｆである；（Ｆ：Ｉｓ→Ｉｓ）。

【００６６】注１）C.E.Shannon. “Comunication theory of secrecy systems” Bell Systems Technical Journal,28(1949), 656-715.

注２）特願平８－１０８０５８

「周期性を考慮したカオス的乱数列の発生装置」及び特願平８－１６９８６９

「乱数生成装置及び乱数生成システム並びに暗号通信方式」

［１８］／／／［００２４］

特願平８－３３８５８３「乱数列の発生方法」

注３）Douglas R. Stinson “CRYPTOGRAPHY：Theory and Practice” Theorem 2.4

注４）　山口昌哉「カオスとフラクタル」ブルーバックス

【００６７】

【発明の実施の形態】請求項１，２，３、４、５、６記載の対称暗号系は、完全守秘性の実現“Perfect Secrecy　Cryptosystem”を意図した。該暗号系を「使い捨てのストリーム暗号システム：One-Time Stream Cipher」としてインターネットへテスト実装した。その実施の形態を述べる。

【００６８】Ｉ、完全守秘性をオープン・ネットワークへ実装する立場から、まず一般的なストリーム暗号システムついて再検討する。

【００６９】図－１にて、ストリーム暗号システムは、平文Ｘをビットストリームであるワーク鍵によって暗号文Ｙに変換する：関数関係式で表すと、

【００７０】

【数１０】

平文Ｘのビットストリーム長＝暗号文Ｙのビットストリーム長、
＝ワーク鍵ｗｋのビットストリーム長、

Bit-length（Ｘ）＝ Bit-length（Ｙ）＝ Bit-length（ｗｋ）　　Eq. 1

暗号文Ｙ ＝ XOR（平文Ｘ＋ワーク鍵ｗｋ）　　Eq. 2

これを元の平文に戻すには、同じワーク鍵ｗｋを使って、

【００７１】

【数１１】

平文Ｘ ＝ XOR（ 暗号文Ｙ＋ワーク鍵ｗｋ ）　　Eq. 3

【００７２】ここで注目しなければならない事は、関数関係式Eq. １&Eq. ２にワーク鍵ｗｋが入っているのでEq. ３の復号が成立するという自明な関係である。情報理論の視点から見ると、一般的に平文Ｘと暗号文Ｙとは互いに「独立な事象ではない」という関係である。これは、そもそも暗号・復号を成立させる自明な、そして必須な関係であることを記憶しておかなければならない。これは対称暗号系の欠点を意味する一方、そもそも暗号系の安全性を計算量の安全性に持ち込まなければならなかった理由でもある。

【００７３】一方、情報理論的な安全性を追求するところの完全守秘性とは「暗号文Ｙが与えられた時に平文Ｘを推定する確率と平文Ｘそのものが生じるアプリオリ確率とは等しい」時の安全性を述べた理論である。この安全性とは、暗号文Ｙが与えられた時に平文Ｘを推定するのは丁度、サイコロの目が“星の数ほど有る”ようなサ

イコロを振って、そしてそのサイコロは誰でも手に入れようとすれば入るものであるが、次にどの“目”が出るか誰も予想する者が居ないということを述べているのと同じである。

【００７４】この完全守秘性は、次のEq. 4とEq. 5が

Prob. P（平文Ｘ｜暗号文Ｙ）＝ Prob. P（平文Ｘ）　Eq. 4

Prob. C（暗号文Ｙ｜平文Ｘ）＝ Prob. C（暗号文Ｙ）　Eq. 5

【００７６】従って、上記条件「平文Ｘと暗号文Ｙとが独立変数である」とは、一般的な関数関係式Eq. １｜２｜３そのものを否定するような主張である。

【００７７】しかし、この時、ワーク鍵が平文Ｘと独立に選ばれ、かつワーク鍵の情報理論的な性質が「どのような平文Ｘ、どのような暗号文Ｙに対してもEq. 2を成立させる鍵が唯一存在し、同時に全ての鍵が等確率に使われるならば、該平文Ｘと暗号文Ｙとは完全守秘性の主張、Eq. 4&Eq. 5を満たす」ということが証明されている：詳細な条件に関しては、注１&注３に譲る。

【００７８】この定理が、どのようにして一般的なストリーム暗号システム（Eq. 1からEq. 3）に実装されたか、以下順に述べるところである。

【００７９】II. まず最初に、当該ストリーム暗号システムの構成要件を図―１に示す；図中に於て、クライアント／サーバー・ネットワーク（以下Ｃ／Ｓネットワーク）などで通信を行う端末双方に、２ｕ（ｕ＞４）進数で演算するレジスターを備え、該レジスターの整数空間Ｉｓのそれ自身への写像（Ｉｓ→Ｉｓ）を実行して整数の軌道を生成する写像アルゴリズムと、２ｕ進数の該カオス的軌道を圧縮変換して２進ビット列の「鍵ストリーム」を生成する圧縮変換部とを備えた鍵生成システムを実装する一方、該端末双方は、該整数の軌道系列の任意の軌道値＜ｃｋ0＞を「秘密鍵」として共有すると同時に、該秘密鍵を初期値として生成した整数の軌道をさらに圧縮変換したところの上記２進ビット列を平文／暗号文と等量作り＜ワーク鍵＞として使う。図中、秘密鍵＜ｃｋ0＞が共有されているので、該秘密鍵から作られたワーク鍵wkも正確に等しい（ The Entropy ＝“0” Bit に示される）。これ等がストリーム暗号システムを構成する（前々特願請求項２、３、に相当する）(注2)。該ス

拡張秘密鍵 ＝ Ｆｉ（秘密鍵）

＜ｃｋｉ＞ ＝ Ｆｉ（＜ｃｋ0＞）

【００８３】Ｆｉ ：該写像アルゴリズムは、レジスターが作る整数空間（Ｉｓ）内の全ての鍵ｃｋ1、ｃｋ2、ｃｋ3、、、ｃｋｉ、、、 ｃｋｎを等確率で生成するだけでなく、該動的ワーク鍵（ビットストリーム）に偏りが無い（0.01%以下）及び該動的ワーク鍵に自己相関がない（0.01%以下）という属性を提供できる非線形関数

One-way Compr：２ｕ進数から２進への一方向圧縮変換

【００８６】該動的ワーク鍵wk（Ｃｋｉ＋ｊ）は、ネットワーク上を移動したり、公開されるような使い方をされる場合も有る、従って盗聴される場合もあるが、該秘

同値であることを主張し、その確率等式は「平文Ｘと暗号文Ｙとが独立変数である」時に成立する：

【００７５】

【数１２】

トリーム暗号システムを構成する枠組みの中で、秘密鍵を情報理論的に安全な拡張秘密鍵“ Secret Key―Enhanced ”にするプロトコルを図―２に示す。

【００８０】該ストリーム暗号システムは、共有している秘密鍵＜ｃｋ0＞を直接使わないで上記鍵生成システムにインプットし、該秘密鍵＜ｃｋ0＞を初期値として写像アルゴリズムを規定回数（ｉ回）実行させて、該秘密鍵＜ｃｋ0＞のカオス的軌道（ｃｋ1、ｃｋ2、ｃｋ3、、、ｃｋｉ）を作らせる。この軌道を該暗号システム内に「保存」した後、このｉ番目のカオス的軌道値＜ｃｋｉ＞を初期値とした写像アルゴリズムを、さらに（ｊ回）実行させ、新たなカオス的軌道（ｃｋｉ、ｃｋｉ+1、、、ｃｋｉ+ｊ）を生成し、当該平文又は暗号文と等量のワーク鍵wkを生成する（ｊの値は平文／暗号文と等量になるように決める）。図―２中、秘密鍵＜ｃｋｉ＞から見て、ワーク鍵wkに至るエントロピーは正確に“0”ビットであり、一方、秘密鍵＜ｃｋ0＞に至るエントロピーは正確に“ｉ”ビットである。故に、秘密鍵＜ｃｋ0＞から見て、該秘密鍵＜ｃｋｉ＞を「拡張秘密鍵」と呼ぶ一方、該ワーク鍵wkから見ると該秘密鍵＜ｃｋ0＞に至るエントロピーはさらに増加し、正確に“ｉ＋ｊ”ビットとなる。これで、より安全なワーク鍵となるので、該ワーク鍵wkを「動的ワーク鍵：Dynamic wk（Ｃｋｉ＋ｊ）」または“secure keystream”として参照する。これら鍵を総称して、カオス的鍵“chaotic keys”として参照する場合がある一方、その生成システムを図―３に示す。

【００８１】該図―３／Ｓ７は、次のEq. ６に対応する；

【００８２】

【数１３】

Eq. 6

でなければならない。“ｉ”は重要なコントロール・パラメータである。

【００８４】該図―３／Ｓ２／Ｓ２／Ｓ５／Ｓ６は、次のEq. ７に対応する；

【００８５】

【数１４】

Dynamic wk（Ｃｋｉ＋ｊ）＝ One-way Compr. Ｆｉ＋ｊ（＜ｃｋ0＞）

Eq . 7

密鍵＜ｃｋ0＞の安全性は「情報理論的に安全である」。ｉ＝１２８として実装すると、該エントロピーは“１２８＋ｊ”ビットである。この不確定度は、サイコロの目が“2^{128} 以上”も有るサイコロを振るのと同じである。

【００８７】該図―１／図―２／図―３に示される構成要件の中で、対称暗号系の秘密鍵を拘束するEq. ６　Eq. ７に示された規約を「安全な同期プロトコル：Secure Synchro―Protocol」として参照する。該安全な同期プロトコルを実装した対称暗号系では、その共通鍵の安全性が従来の計算量の安全性の問題“Computational Security”にならず、情報理論的に安全な又は確率論的に安全な問題“Unconditional Security”に帰着する。その理由を要約すれば、

【００８８】①　該安全な同期プロトコル“ Secure Synchro―Protocol：ＳＳＰ ”という実装技術が、端末群に共有され、

②　端末双方のカオス的鍵“chaotic keys”を制御し、

③　その秘密鍵＜ｃｋ0＞を情報理論的に安全な拡張秘密鍵＜ｃｋｉ＞に変換し、これから動的ワーク鍵 wk（Ｃｋｉ＋ｊ）“secure keystream”を生成する、からである。

【００８９】III.この安全な同期プロトコル（ＳＳＰ）をパラダイムにして、秘密鍵をどのようにして“One-Time”にするかという本題に入ることになるが、その前提として、鍵をどのようにしてネットワーク上を配送するか、そのためにはクライアント／サーバー双方の認証をどのように実施するか、を示さねばならない。「代理鍵の安全同期配送プロトコルおよび代理鍵による安全同期認証プロトコル」図―４と「初期認証シーケンス／Passnet更新シーケンス」図―６はその仕組みを示す。ＳＳＰのパラダイムでは、認証と鍵配送とが同時進行し、内容は同じであるが違った目的に到達するという特徴がある。

【００９０】その構成要件は図―２に示すように、秘密鍵＜ｃｋ0＞を共有している請求項１、２記載のストリーム暗号システムは、Ｃ／Ｓとも共有している秘密鍵＜ｃｋ0＞を、互いに相手を確認しうる手段であるＩＤ＝秘密鍵をネットワーク上で交換する事をせず、両者とも「保持する」。

【００９１】その代わり、図―４、図―５に示すように、該ストリーム暗号システムはサーバー上で他の偶然的イベントをトリガーとした初期値から別のカオス的軌道値＜ｓｋｎ.serv＞を生成し、これをサーバーが保持している秘密鍵＜ｃｋ0.serv＞の「代理鍵」として、前記２項の安全な同期プロトコルと該秘密鍵＜ｃｋ0.serv＞によって暗号化し、クライアントに配送する。この偶然的イベントの発生は、いままで知られているどのような手段でもよい；「代理鍵」は一度使用されるとその都度、瞬時に捨てられるから。

【００９２】配送の様子を図―６にて、１―秘密鍵ＤＢから７―暗号化された“ｓｋｎ”までに示す。１―秘密鍵ＤＢから該当するユーザーの秘密鍵＜ｃｋ0.serv＞をどのように探し当てるか、この仕組みを「初期認証」というが、これには“Passnet”と名付けられたパスワードを使う。“Passnet”は、ユーザーが携帯するメデイア（図―６―９に示す）に、ユーザーの秘密鍵＜ｃｋ0.serv＞と一緒に格納されているランダム・データである。

【００９３】該“Passet”とは、どのようなパスワードか後述する（なおユーザーの携帯するメデイアにどのように「安全に格納する」かの仕組み等は派生技術なので除外する）この暗号データを受け取ったクライアントは（図６では初期認証応答として示される）、該端末が保持していた秘密鍵＜ｃｋ0.clin＞と請求項１記載の「安全な同期プロトコル（ＳＳＰ）」により同じく該暗号データを復号化して代理鍵＜ｓｋｎ.clin＞を受け取る。

【００９４】但し、この段階では「サーバーが保持するクライアントの秘密鍵」と「クライアントが保持している秘密鍵」とは等しいとは限らないので、前者を＜ｃｋ0.serv＞後者を＜ｃｋ0.clin＞と記号して区別している（図―４では＜Ｓｋ？＞と表現している）。

【００９５】本来、秘密鍵は共有されている一対のＩＤ兼用暗号鍵であるが、ネットワーク上でパスワードで引き当てただけでは、それが後述する“One-Time Password”であっても、両者の秘密鍵が同じであるとは保証できない。

【００９６】従って、この時点では秘密鍵、代理鍵ともクライアントに所属していたものかサーバーに所属していたものなのか、両者の区別を記号化した；図―６では、＜ｓｋｎ.serv＞を７―暗号化される代理鍵、一方＜ｓｋｎ.clin＞を１２―復号化された代理鍵として区別している。

【００９７】このような構成要件の中では、秘密鍵が同じであれば代理鍵も等しいし、その逆に、代理鍵が等しければ秘密鍵もまた等しいことが判る。故に認証のためには秘密鍵の代わりに代理鍵を調べる。もし該暗号化される代理鍵と復号化された代理鍵とが等しいことが、安全に確認出来たとすると、

【００９８】

【数１５】

$$＜ｓｋｎ.serv＞ ＝ ＜ｓｋｎ.clin＞ ＝ ＜ｓｋｎ＞ ：　　Ｅｑ．８$$

【００９９】結局、鍵を共有するようになるので、該ストリーム暗号システムは、鍵の情報理論的に安全な配送を「認証と同時に」完了する「鍵の同期配送プロトコル」を備えた事になる。

【０１００】IV.該Ｅｑ．８を確認する手段を「鍵の同期認証プロトコル」と名付け、以下の図―７と図―４に

説明する。代理鍵を直接ネットワーク上で交換するのは“insecureとなるから、代わりに、図―７に於いてサーバー上の代理鍵１５＜ｓｋｎ.serv＞を初期値としたカオス的軌道１６―

【０１０１】

【数１６】｛ｓｋ（ｎ＋１）.serv、ｓｋ（ｎ＋２）.serv、ｓｋ（ｎ＋３）.serv、、、｝

【０１０２】を生成し、一方クライアント上の代理鍵１７＜ｓｋｎ.clin＞についても、それを初期値としたカオス的軌道系列１８―

【０１０３】

【数１７】｛ｓｋ（ｎ＋１）.clin、ｓｋ（ｎ＋２）.clin、ｓｋ（ｎ＋３）.clin、、、｝

【０１０４】を生成し、これらカオス的軌道系列を前記乱数生成システムに戻して該圧縮変換部に入力し、それぞれ対応するワーク鍵系列に変換する；

【０１０５】

【数１８】（、、ｗｋ１.clin、ｗｋ２.clin、ｗｋ３.clin、、、）；　図―７―２２／２３／２４

【０１０６】及び

【数１９】（、、ｗｋ１.serv、ｗｋ２.serv、ｗｋ３.serv、、、）；　図―７―１９／２０／２１

【０１０７】これらワーク鍵系列は「情報理論的に安全な拡張ワーク鍵群」である。従って代理鍵を直接ネットワーク上で交換する代わりに、その拡張ワーク鍵をネットワークへ出して交換する事が出来るようになる。該拡張ワーク鍵を初期認証に使う場合、図―６では“Passnet”と呼び、通常の“Password”と区別した。交換のプロトコルは比較的に新しい「拡張ワーク鍵」＜ｗｋ３.clin＞を先に使う（この理由も既に述べた）。

【０１０８】すなわち、

（１）クライアントからサーバーへ最も新しい「拡張ワーク鍵」＜ｗｋ３.clin＞を配送する―２５。

【０１０９】（２）サーバー上では２６―下記の比較をして一致していれば「クライアントの認証」を行う：クライアントのワーク鍵＜ｗｋ３.clin＞：サーバーのワーク鍵＜ｗｋ３.serv＞サーバーがクライアントの認証を実施した場合、その確認のサインをどのような手段でクライアントへ通信するか、通常のＴＣＰ／ＩＰのプロトコルを使うのは“insecure”となるのは明らかである。そこで、

（３）サーバからクライアントへも「拡張ワーク鍵」を配送する。この鍵には拡張ワーク鍵であれば＜ｗｋ１.serv＞又は＜ｗｋ２.serv＞どちらを使ってもよい。

【０１１０】図―５／２では、＜ｗｋ１.serv＞を使っている―２１。

【０１１１】（４）クライアント上では―２６の比較をして次のことを確認する：

①クライアントはサーバ上でクライアントの認証が実施された事を確認する；

②クライアントがサーバを認証する；

（５）一致を見なかった場合は、該端末の方のイニシャティブで通信を切断する―２７。

【０１１２】この鍵の同期認証プロトコルでは、

【０１１３】

【数２０】

拡張ワーク鍵の一致　：＜ｗｋ３.serv＞＝＜ｗｋ３.clin＞
＝＜ｓｋn.serv＞＝＜ｓｋｎ.clin＞：代理鍵の共有し
＝＜ｃｋｉ.serv＞＝＜ｃｋi.clin＞；拡張秘密鍵の共有し
＝＜ｃｋ0.serv＞＝＜ｃｋ0.clin＞；秘密鍵の共有し
＝認証完了（図―４に示す）　　Ｅｑ．９

【０１１４】そして、同時に前述のＥｑ．８を成立させる；

【０１１５】

【数２１】

＝代理鍵の情報理論的に安全な配送
＝ｓｋｎ：代理鍵を共有する；　　Ｅｑ．８

【０１１６】故に「鍵の同期配送プロトコル」と「鍵の同期認証プロトコル」とは同じ手続きに基ずく二つの異なる適用であることが判る。これはカオス的写像の安全な同期制御に基ずく、あるいは安全な同期プロトコル（ＳＳＰ）に基ずく鍵共有の新しい方式を与える。「鍵の同期配送プロトコル」と「鍵の同期認証プロトコル」の両者を合わせて、このスキームを「認証付き鍵共有プロトコル：Authenticated Key Agreement Protocol（AKAP）」として参照する；すなわち、

【０１１７】

【数２２】AKAP ＝ SSTP ＋ SSAP

SSTP：Secure Synchro-Transfer Protocol

SSAP：Secure Synchro-Authentication Protocol

【０１１８】該認証付き鍵共有プロトコルは種々の目的に応用出来る。これをサーバーの視点で考えると、従来に無い優れた特徴を備えた「鍵サーバー：key server」の機能を与える。

【０１１９】Ｖ．“One-Time Password”；代理鍵とそのワーク鍵に関する安全な同期制御（ＳＳＰ）よれば、

【０１２０】

【数２３】Ｅｑ．８／Ｅｑ．９＝その他の拡張ワーク鍵群を共有する、Ｅｑ．１０

【０１２１】も同時に実現する。該「その他の拡張ワーク鍵」―Ｅｑ．１１も共通鍵であることに着目すると、

次回クライアトがサーバーへ“loginする際のパスワードとして使える事を発見する。しかも該拡張ワーク鍵は、代理鍵の代わりをする、情報理論的に安全な「公開鍵」であるだけでなく、クライアントがサーバーへ“loginする度に、該サーバー上で他の偶然的イベントから生成した整数の軌道値であったことを思い出すと、該拡張ワーク鍵が“One-Time”であることは明らかである。

該拡張ワーク鍵を“One-Time Password”にする手順

＜ｗｋ０＞　←　＜ｗｋ２＞

【０１２４】該ワーク鍵＜ｗｋ２＞が次回使われる“Passnetである。

【０１２５】こんな簡単なプロトコルで“One-Time Password”が成立するのは、前記の認証付き鍵共有プロトコルのお陰である：図―８／検証―１に示されるシステムはインターネット上に公開されているので、実際に鍵サーバーへ“loginを実施して“One-Time”検証出来る；“loginをインターネット上で連続して実施し“One-Time”の“Passnetを検証する；（http：//ｗｗｗ.meteora.amadagp.co.jp/）。

【０１２６】VI.“One-Time”のセッション鍵系列；ここで共有された代理鍵は、種々のアプリケーションを可能にする；Ｅｑ．８より

【０１２７】

【数２５】＜ｓｋｎ.serv＞＝＜ｓｋｎ.clin＞＝ｓｋｎ：ｎ＝0～ｉ

【０１２８】これは拡張共通鍵群であるから、

① 該代理鍵はクライアントとサーバーが暗号セッションを持ったという証拠に使える。これは、商取り引きの事実を証明する、保存に値するデータである：“Non-repudiality in Electric Commerce。

【０１２９】② 該クライアント・サーバー双方は、安全な同期プロトコル（ＳＳＰ）を介して“One-Time”のセッション鍵を「等確率に」、「連続して」、「ほとんど無数に」共有する。該拡張共通鍵群を多数共有している様子を図―５に示す。該“One-Time”のセッション鍵を、ｗｋ２“Passnetの更新処理；図―７―２８／２９に示す。

【０１２２】図―６―９のデイスクに示される“Passnetとは、初めの“loginに使われた＜ｗｋ０＞である；これを図―７―２８／２９で、新しい拡張ワーク鍵＜ｗｋ２＞と入れ替える：

【０１２３】

【数２４】

Ｅｑ．１１

の使い道には、例えば、特定のクライアントだけに特定のコンテンツやバリューデータを提供するためのＩＤの機能を持たせるアプリケーションが有る。（なお、図―９／検証―２のシステムはインターネット上で検証出来る：（http：／／ｗｗｗ.meteora.amadagp.co.jp／を参照ください）。三番目に、

③ 該サーバー（鍵サーバー）に参加しているクライアント（Ａ）と他のクライアント（Ｂ）に共通のセッション鍵を安全に与えることが出来る。一度設定されると、該鍵サーバを介さないでも、クライアント（Ａ）―（Ｂ）は“One-Time”のセッション鍵系列を「ほとんど無数に」共有する。このアプリケーションは従来の“Kerberos”などに見られない便宜をユーザーに与える。

【０１３０】④ 鍵サーバー、セッション鍵、該セッション鍵から作った“Passnetなどが“One-Time”のデジタル署名技術を構成する。

【０１３１】これら具体的な適用事例の紹介は、認証付き鍵共有（AKAP）の派生技術であるので、省略し請求から外した。

【０１３２】VII.“One-Time Stream Cipher”；安全な同期プロトコル（ＳＳＰ）をベースにした「認証付き鍵共有」を成した後に於いては、当該ストリーム暗号システムの中の拡張秘密鍵は

【０１３３】

【数２６】

拡張秘密鍵の一致　＝＜ｃｋｉ.serv＞＝＜ｃｋｉ.clin＞；ｉ＝1～任意

＝＜ｃｋ0.serv＞＝＜ｃｋ0.clin＞；

＝＜ｃｋ0＞；秘密鍵を共有する；　　Ｅｑ．９

【０１３４】を無条件に満足する。この事は、「ＴＣＰ／ＩＰなどの一般的なプロトコルを介しても」、認証付き鍵共有を経過することにより、「写像アルゴリズムの安全な同期制御」をリモートで正確に実施できると同時に、システムを“fail-safe”に設計出来ることを意味する。

【０１３５】よって、“login”する度に、拡張秘密鍵をもって秘密鍵を正確・確実に更新出来る手段を得ることになった。該拡張秘密鍵の一つ＜ｃｋ１＞（図―２に示される）は、そのエントロピーが“ｉ―１”ビットであるので、元の秘密鍵＜ｃｋ0＞の次に情報理論的に安全な鍵である。よって秘密鍵の更新処理は極めて単純となる：

【０１３６】

【数２７】

＜ｃｋ0＞　←　＜ｃｋ１＞

Ｅｑ．１２

【０１３７】該更新のタイミングは、前述図―７―２８／２９の“Passnetの更新処理と同じ である。この様な単純なプロトコルの適用で対称暗号系を“One-Time”に出来るのは、該暗号系が認証付きの鍵共有プロトコルによって成立しているからである。

【０１３８】VIII. 鍵管理と鍵サーバー；オープンネッ

トワークでは、一般に第三者の信頼できる機関（trusted authority:TA）または証明機関（certificate authority：CA）を必要としている。ユーザの身分を証明したり、ユーザの鍵を送付したりする。鍵を管理するという事は、これだけの適用に止まらない。鍵を生成し管理するというこのTA、CAこそ“データに価値を与える”要である。

【０１３９】つまり署名技術の核心とその適用を見抜くならば、TA、CAを第三者の“信頼できる人”に期待するのではなく、“第三者の信頼できるシステム”にしなければならないことが判る。丁度、紙幣や小切手を人手に頼ってガリ版で刷っているような状況を想像してみれば良い。

【０１４０】請求項５の発明は、鍵の運用管理に関してクライアント／サーバーという当事者以外に一切人を介在させないこと、及び鍵を情報理論的に安全なセキュリテイで守る仕組みに関する。サーバーは多数のクライアントの秘密鍵のファイル「鍵箱」を備える；図―６、秘密鍵ＤＢがそれに該当する。秘密鍵ＤＢの管理を次のように行う；サーバーは、前１、２項記載の安全な同期プロトコル（ＳＳＰ）の下で前３項の「認証付き鍵共有」を成した後に於いて、鍵箱の任意の秘密鍵＜ｃｋ0＞を前１、２項記載の当該秘密鍵＜ｃｋ0＞の「鍵ストリーム」＜ｗｋ-ｃｋ0＞によって該鍵＜ｃｋ0＞自身を暗号化する。従って、該「鍵箱」は暗号ファイルCRYPTO（ｃｋｎ：ｎ＝0,1,2,3,,）として管理される一方、該サーバーは「復号化の手段を失う」。

【０１４１】該復号化の手段を持っているのはクライアントだけである。すなわち、クライアントが保持している鍵＜ｃｋ0.clin＞から生成したところの前１、２項記載の「鍵ストリーム」、＜ｗｋ-ｃｋ0.clin＞しか復号化の手段が該システムに残されていない。そこで、前記１乃至４項の対称暗号系は次の“login”時に、クライアントの該鍵ストリーム＜ｗｋ-ｃｋclin＞をサーバーに配送し、該サーバー上で暗号ファイル CRYPTO（ｃｋｎ：ｎ＝0）を復号化し、当該クライアントの秘密鍵＜ｃｋ0.serv＞をサーバーへ与えるとした「鍵管理アルゴリズム」を実装する：

（１）初期認証時（図―６―１、９→１）に於いて、“passnetの送付と共に、上記鍵ストリーム＜ｗｋ-ｃｋ0.clin＞を送付する。

（２）該サーバーは、該秘密鍵＜ｃｋ0.serv＞を入手する。

【０１４２】ここで、暗号化以前の＜ｃｋ0＞と復号化後の秘密鍵＜ｃｋ0.serv＞とは必ずしも等しくないことに注意する。

【０１４３】ここからは、先のIII―代理鍵の安全同期配送プロトコル、及びIV―代理鍵による安全同期認証プロトコルを実施する。言い換えると、前３項記載の規約［認証付き鍵共有」を成して前記１乃至４項記載の対称暗号系を成立させるのである。

【０１４４】このように、極めて単純なアルゴリズムで、クライアント／サーバーという当事者以外に一切人を介在させない鍵の運用管理及び鍵サーバーを実現出来るようになる。

【０１４５】IX.フラクタルな演算構造

請求項１及び６記載の鍵生成システムは、カントール集合を生む従来のフラクタル構造以外に、有限精度のコンピュータの中に導入された新たなフラクタルな演算構造を持つ。その結果を図―１０、図－１１に示す。この事例では、非線型関数に、ロジステイックス曲線を使っている。該自己相似形レジスターの仕様は：

一様レジスターの桁数　；１０００個：10^3、

入れ籠レジスターの桁数；１０００個：10^3、

一様レジスター上の演算は、10^3を“法”とする。

【０１４６】該自己相似形レジスターにより、連続量である線分［０、１］が“discreteな１０００個の微少区間に分けられ、その微少区間がさらに１０００個の微少区間に分けらるという風に順次考えることが可能となる。その結果連続量の属性の一つである［加算個の無限］を“discreteな自己相似形アルゴリズムで扱えるようになる。この時のフラクタル次元は従来の“0.６３”から“0.63+1.0”に上がる（該理論は他に譲る、ここではその様子を実施例、図―１０、図－１１で示した）。図―１０のプロットは、１５００回程度の写像演算をした結果の、一様レジスターの先頭桁の数値を表す；これが、ロジステイックス曲線を意味することは明らかである。同時に、図－１１は、該先頭桁の次の桁の数値を表すプロットである。これ以降の桁の数値も、全て該図－１１のプロットのような様相を示す（省略する）。この乱雑なプロット図－１１と先の整然としたプロット図―１０の意味するところは、乱雑な方のプロットが“Lorentz plot”注[4]に相当する。一見乱雑に見える現象（図－１１）と（例えば、お湯の沸騰状況がこのような“Lorentz plot”として現れるが）、その現象の背後には整然とした法則性が存在しているという関係、この場合ロジステイックス曲線（図―１０）が存在しているという関係を、プロット図－１１と図―１０が示したのである；一見乱雑に見える現象（図－１１）とその背後にある法則（図―１０）とのシミュレーションである。

【０１４７】請求項６記載のフラクタルな演算構造とカオス的写像アルゴリズムが、カオス的軌道を成すという状況証拠としては、図―１０＆図－１１以外に、鍵ストリームの自己相関が0.01%以下、１／０ビットの偏りも0.01%以下というデータに現れている。

【０１４８】

【発明の効果】発明の実施の形態に於いて、“One-Time Stream Cipher”をオープン・ネットワークの完全守秘性暗号系“Perfect Secrecy Cryptosystem”として紹介

した。

【０１４９】従来、“one-time pad cipher”は完全守秘性暗号系であるにもかかわらず、オープン・ネットワークへ適用出来なかった。“One-Time Stream Cipher”はその困難を解決した。その発明された問題解決法とは、一つは情報理論的に安全な鍵生成システム“Secure Key Stream”と情報理論的に安全な同期プロトコル“Secure Synchro—Protocol”である。その上に築かれている、もう一つの発明が、認証付き鍵共有プロトコル“authenticated key agreement protocol”である。実は“ One-Time Pad Cipher”に必要であった「もう一つ別の安全な通信路」とは、この鍵共有プロトコルがその仕組みを提供しているのである。例えば、インターネット上で“One-Time Pad cipherを実現するという事は“One-Time Padcipherと同等な安全性を帯びている「認証技術」も同時に備えねばならないことを意味している。しかし、ここにＲＳＡは使えない；この安全性は計算量の安全性“Computational Security”であるから。

【０１５０】従って、当該発明はインターネット上で使える情報理論的に安全な認証技術を発明したという主張に等しい；それをストリーム暗号系に実装し、“One-Time Stream cipher”と名ずけたのである。

【０１５１】この視点から見ると、認証付き鍵共有プロトコルを実装したサーバーは「情報理論的に安全な鍵サーバー」である。もし、従来のプロキシーサーバーに該鍵サーバーを実装したらどうなるか；該鍵サーバーは使用回数が無制限の“One-TimePassword”をサービスする。同じく、ｗｗｗサーバーに実装すれば、該ｗｗｗサーバーは、①認証機能“One-Time Password”を備え、②イントラネットのＤＢを暗号ＤＢに成し、その暗号と復号の機能を提供し、③インターネットとイントラネットをシームレスに統合する新しいオープンネットワークを提供する。

【０１５２】さらに、該鍵サーバーの下に金融証券のアプリケーションを置いたらどうなるか。当然、該金融機関が該鍵サーバーを管理するから、該システムは、従来の金融機関の基本システムを維持したまま、しかもオープン・ネットワーク上でのトレードや決済を可能にする。将来、金融システムにも適用可能な暗号系とは、このような鍵サーバーを含んだ完全守秘性暗号系になるはずである。

【図面の簡単な説明】

【図１】ストリーム暗号の生成を説明する説明図である。

【図２】拡張秘密鍵と情報理論的に安全なストリーム暗号システムを説明する説明図である。

【図３】カオス的鍵と鍵ストリームの生成・更新を説明するフローチャートである。

【図４】代理鍵の安全同期配送プロトコルと代理鍵による安全同期認証プロトコルを説明する説明図である。

【図５】Ｏｎｅ－ｔｉｍｅのセッション鍵群を説明する説明図である。

【図６】初期認証シーケンスを説明する説明図である。

【図７】更新シーケンスを説明する説明図である。

【図８】検証１を説明する説明図である。

【図９】検証２を説明する説明図である。

【図１０】ロジステック曲線の説明図である。

【図１１】本発明による演算結果を説明する説明図である。

【符号の説明】

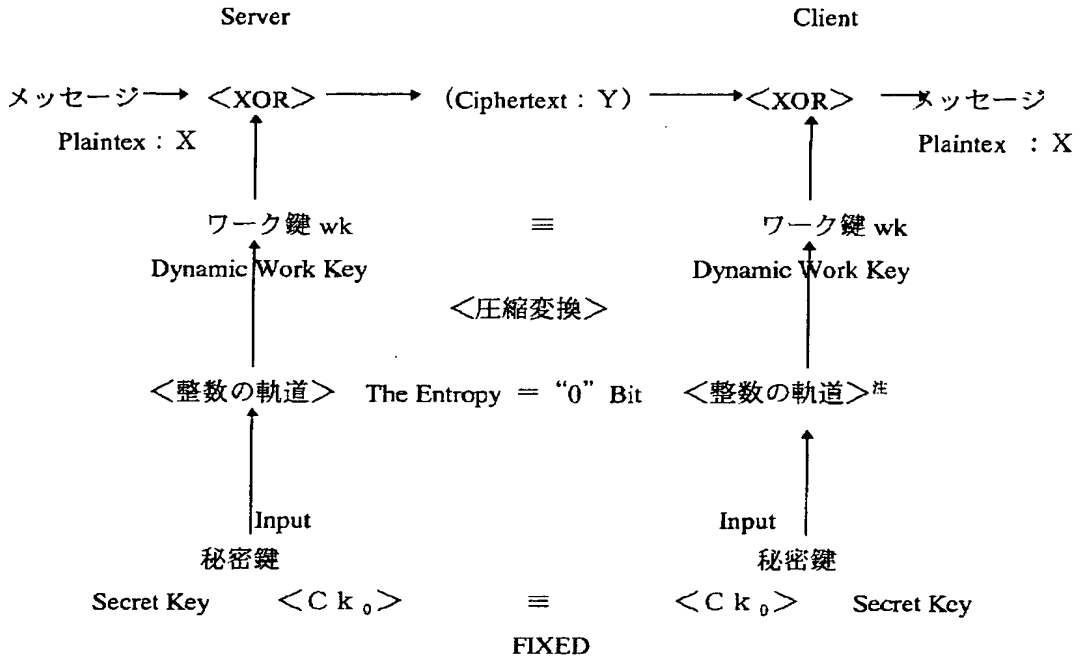
Ｓｋｏ　セッション鍵

Ｗｋ　ワーク鍵

【図１】

ストリーム暗号

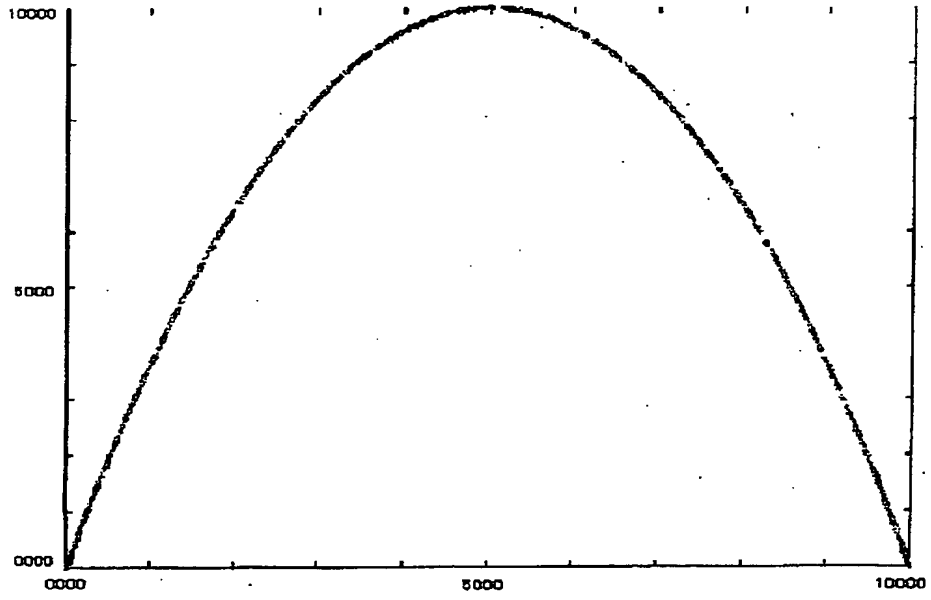
Stream Cipher



注）

<整数の軌道>： 秘密鍵<C k $_0$> を初期値とした軌道を指す：

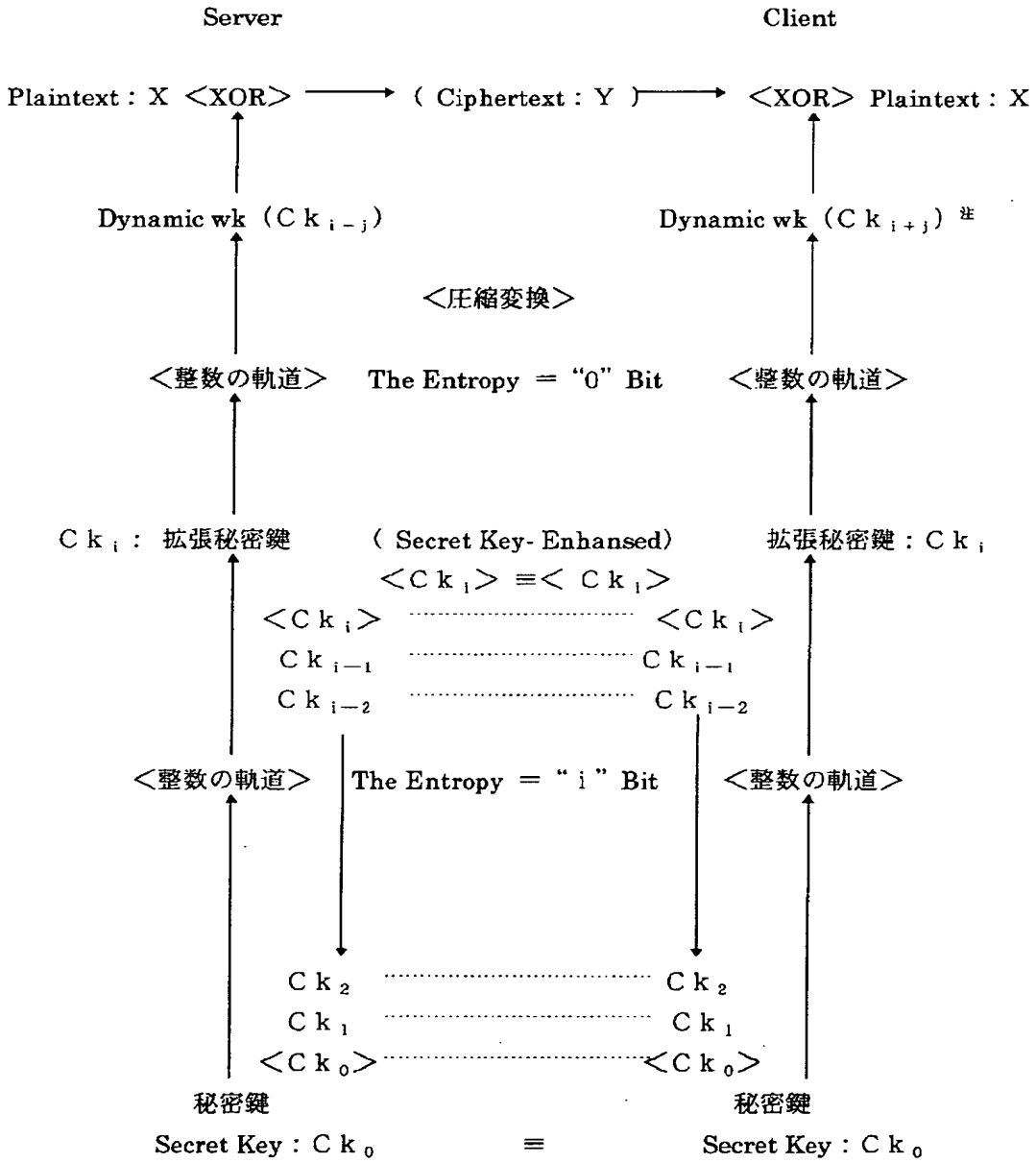
【図１０】



【図２】

拡張秘密鍵と情報理論的に安全なストリーム暗号システム

＜ Secret Key_Enhansed and Unconditionally_Secure Stream Cipher System ＞

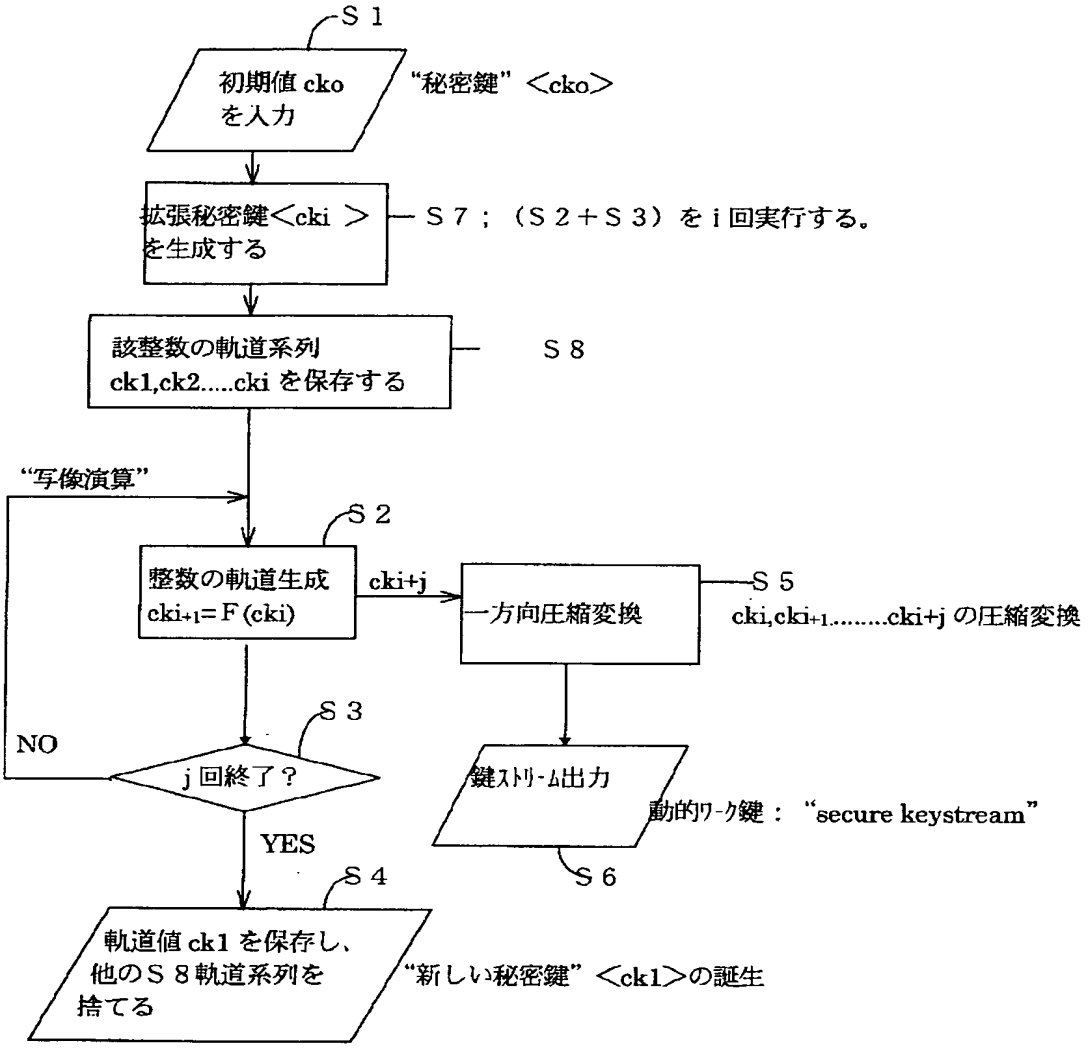


注)

Dynamic Wk（Ck $_{i+j}$）：＜Ck $_i$＞を初期値として新たに生成した整数の軌道ΞCk $_{i-j}$を圧縮変換したところのビットストリームである。

【図３】

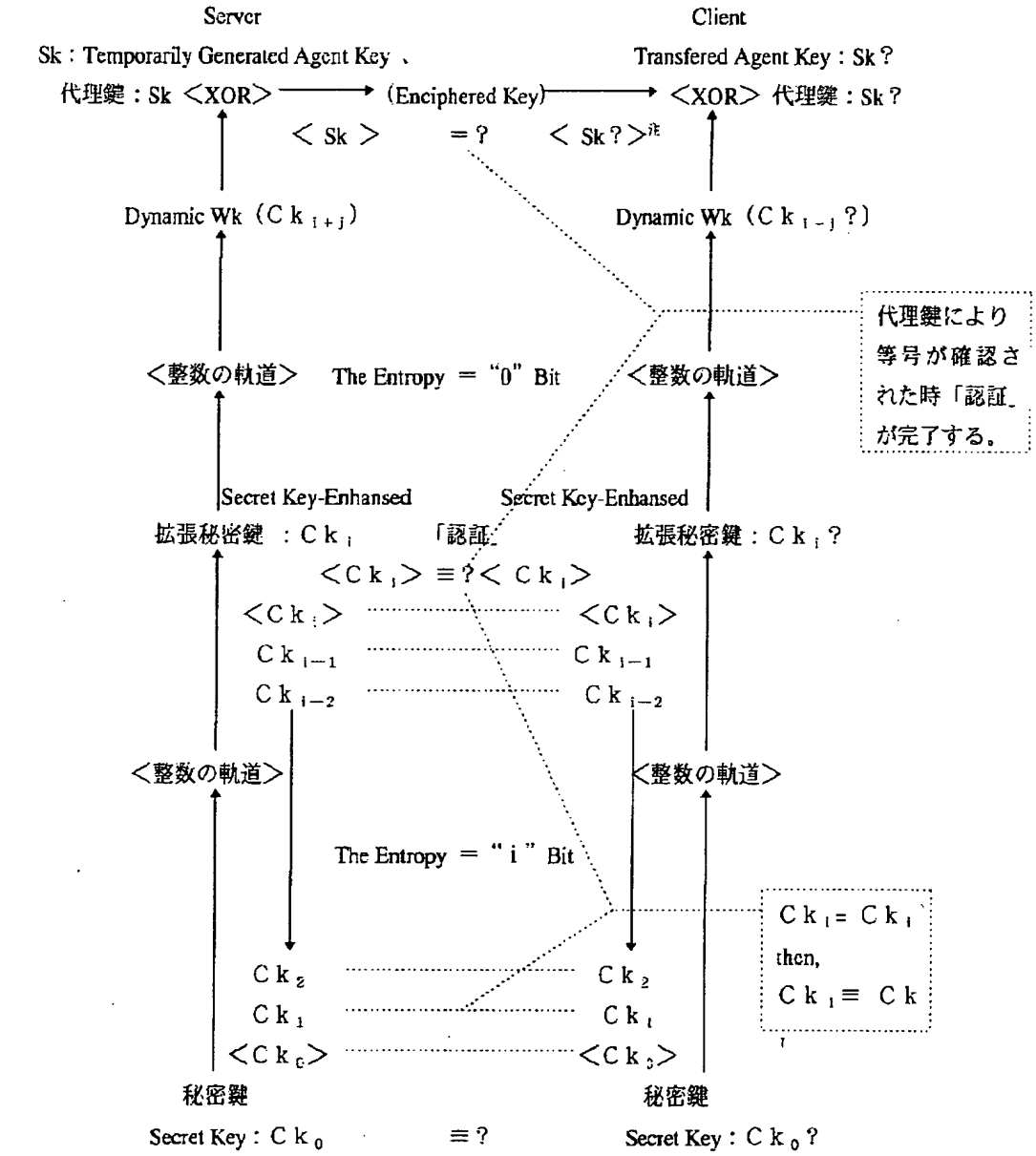
カオス的鍵と鍵ストリームの生成・更新のフロー



【図４】

代理鍵の安全同期配送プロトコルと代理鍵による安全同期認証プロトコル：

＜ Secure Synchro-Transfer Protocol For Agent Key and
Secure Synchro-Authentication Protocol by Agent Key ＞



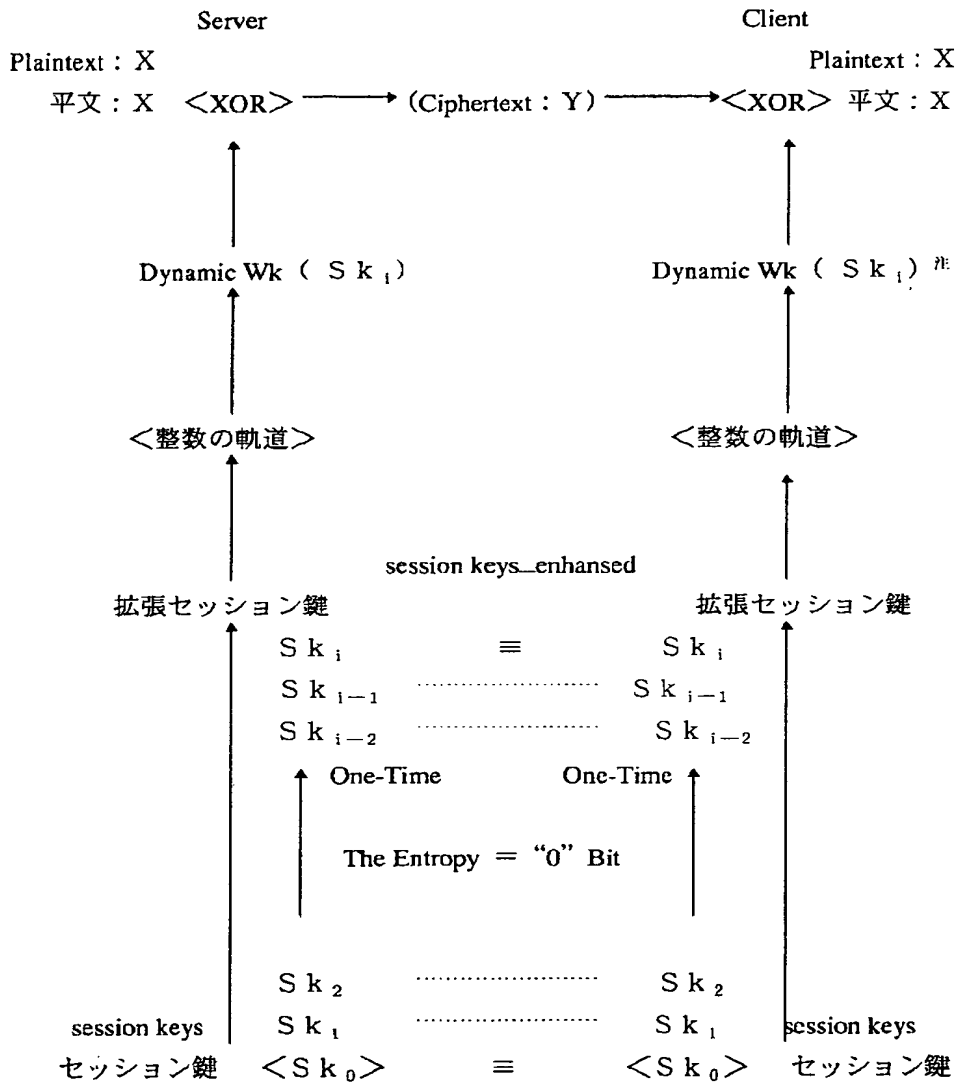
注）

＜ Sk. serv ＞ ＝ ＜ Sk. clin＞, then ＜Ck i. serv＞ ≡ ＜ Ck i. clin＞
＜Ck i. serv＞ ≡ ＜Ck i. clin＞

【図５】

"One-Time" のセッション鍵群

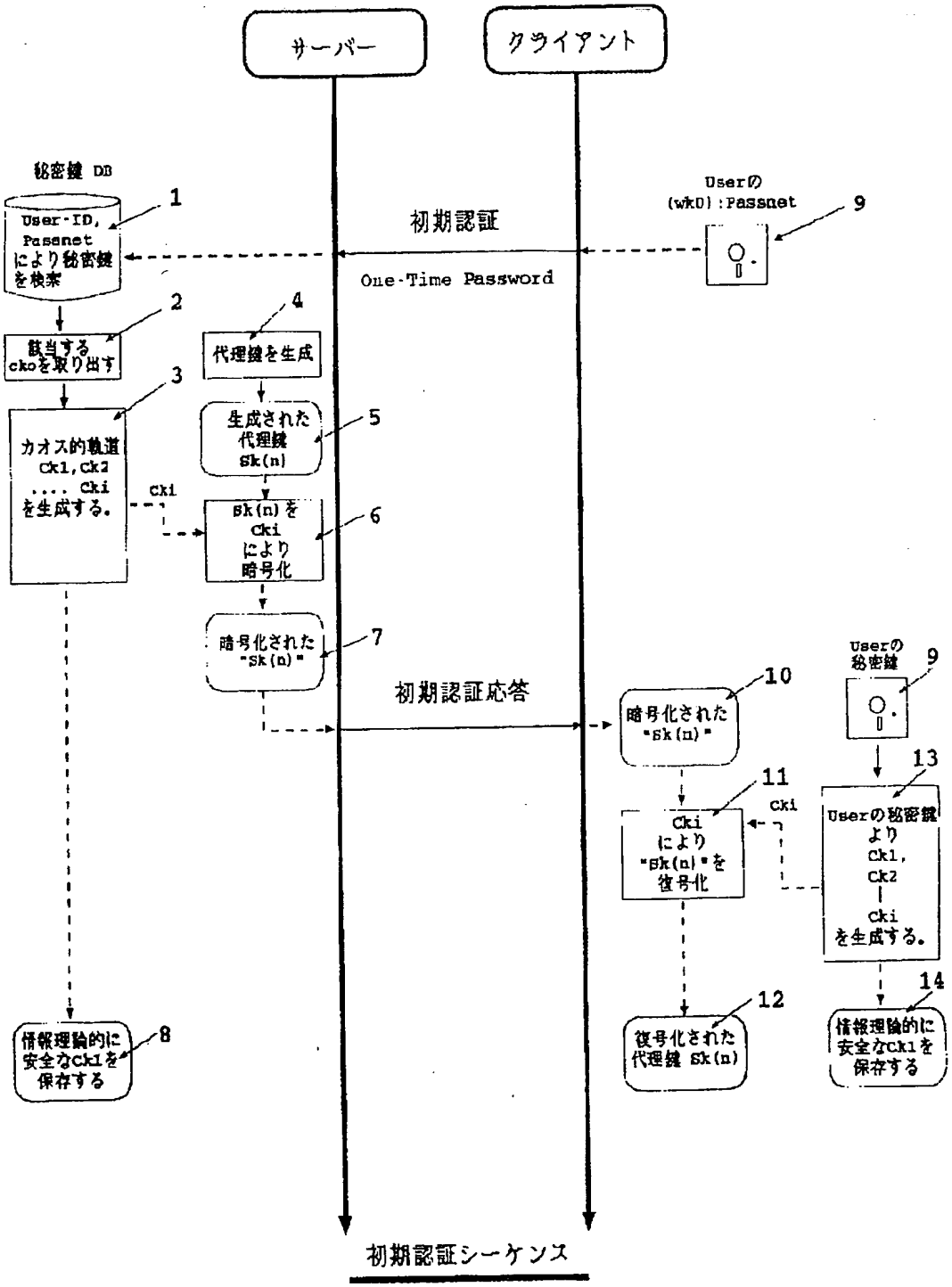
< one-time session keys >



注)

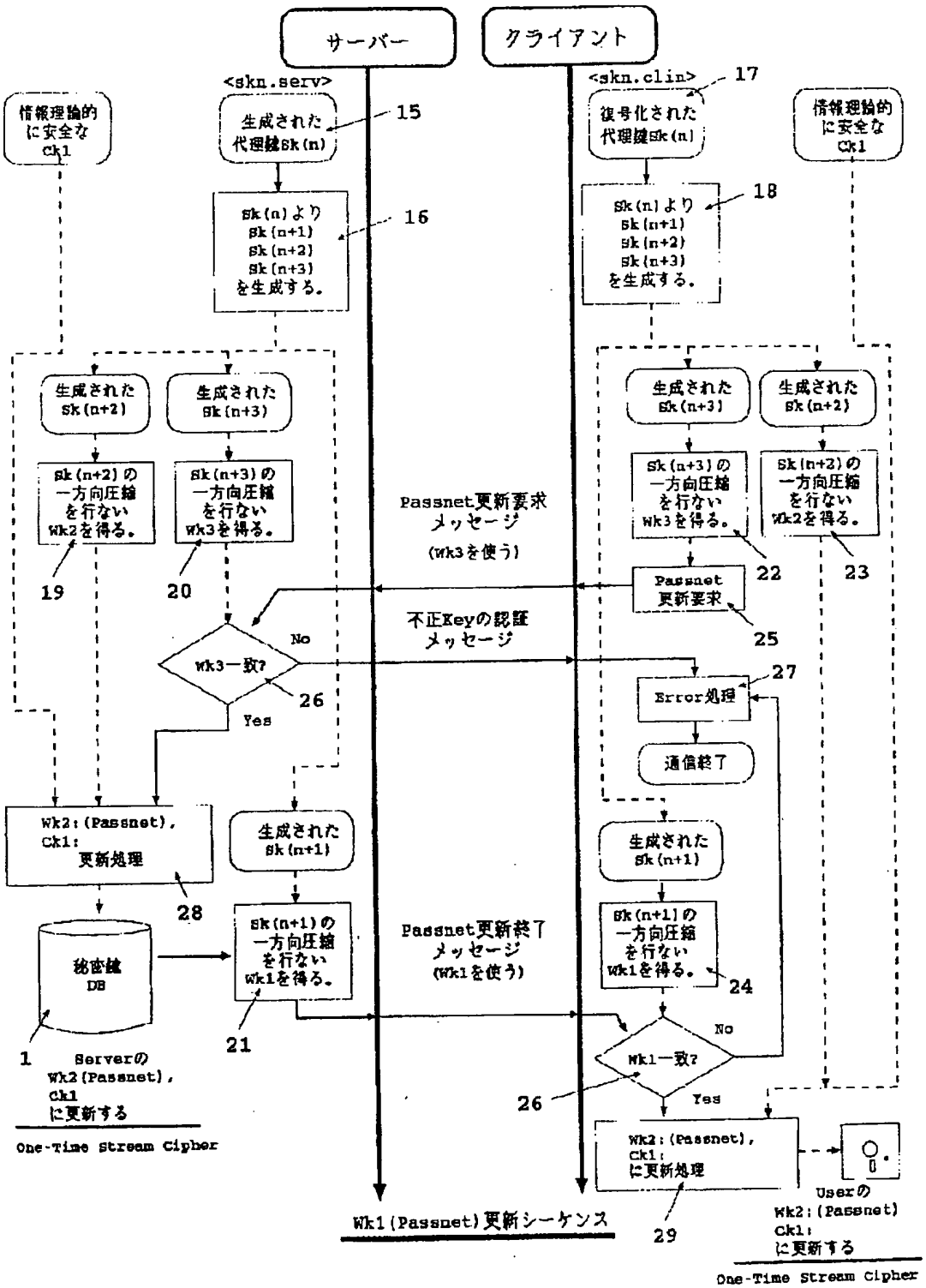
Dynamic Wk (Sk i) : (Sk i、 i＝1)、(Sk i、 i＝2)、(Sk i、 i＝i) をそれぞれ初期値としたところの "i" 本のビットストリームを指す：

【図６】

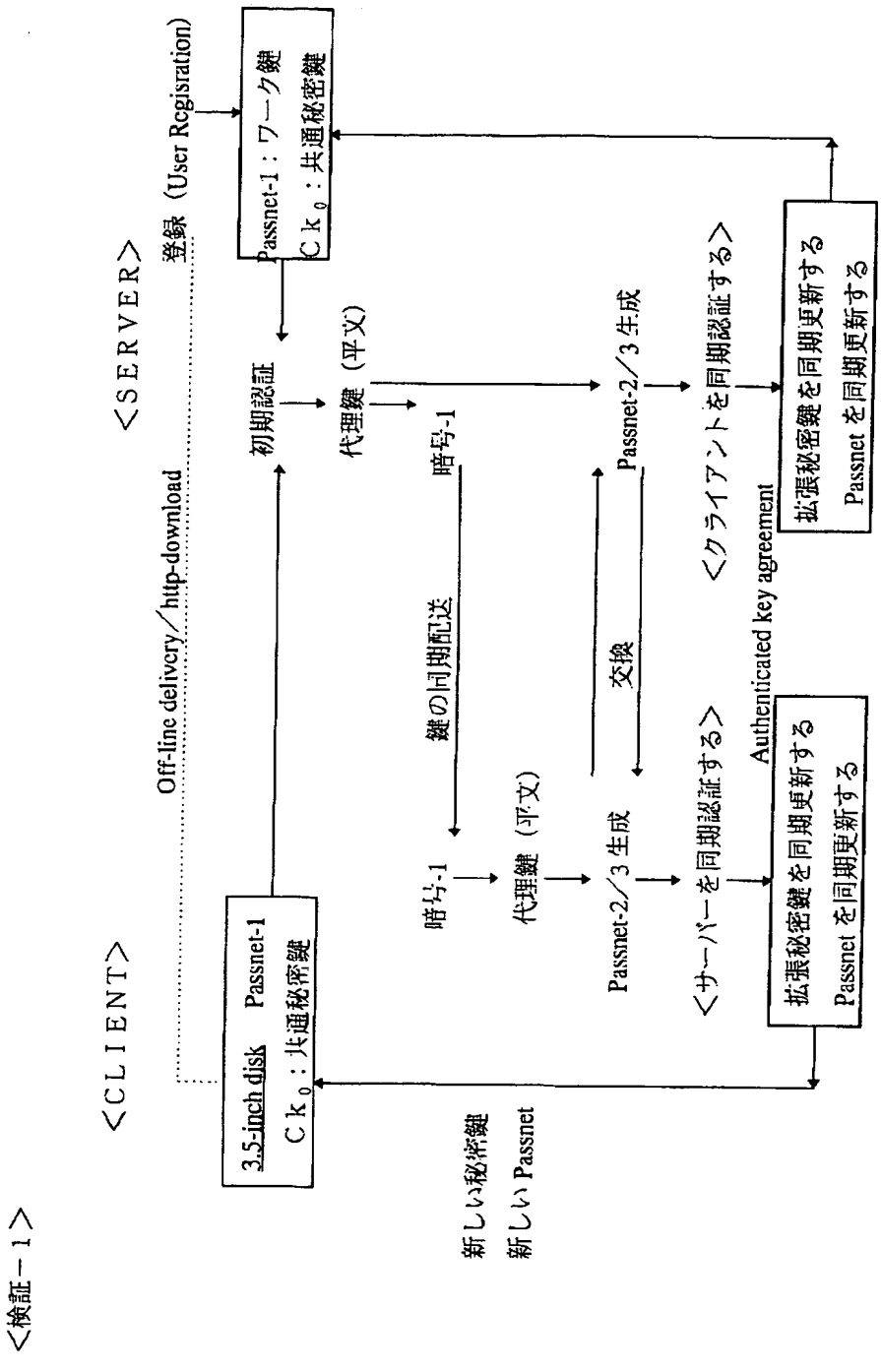

サーバー
クライアント
秘密鍵 DB
User-ID, Pasenet により秘密鍵を検索
1
初期認証
One-Time Password
Userの (wkD):Passnet
9
該当する ckoを取り出す
2
代理鍵を生成
4
カオス的軌道 Ck1,Ck2 .... Cki を生成する。
3
生成された 代理鍵 Sk(n)
5
Cki
Sk(n)を Cki により 暗号化
6
暗号化された "Sk(n)"
7
初期認証応答
暗号化された "Sk(n)"
10
Userの 秘密鍵
9
Cki により "Sk(n)"を 復号化
11
Cki
Userの秘密鍵より Ck1, Ck2 | Cki を生成する。
13
情報理論的に安全なCk1を保存する
8
復号化された 代理鍵 Sk(n)
12
情報理論的に安全なCk1を保存する
14
初期認証シーケンス

Cki;拡張秘密鍵

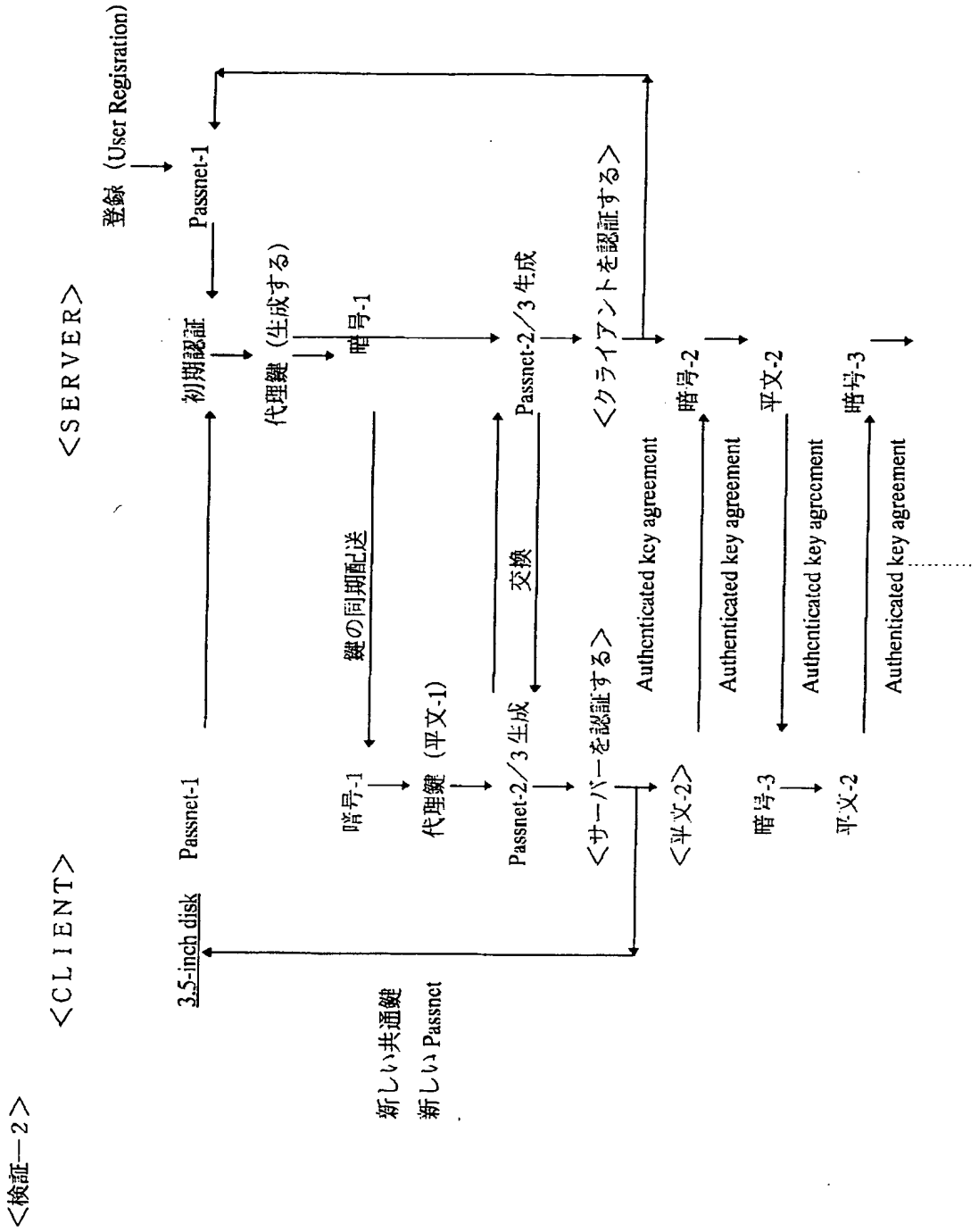
【図７】


サーバー
クライアント
<skn.serv>
<skn.clin>
情報理論的に安全なCk1
生成された代理鍵Sk(n)
復号化された代理鍵Sk(n)
Sk(n)より Sk(n+1) Sk(n+2) Sk(n+3) を生成する。
生成されたSk(n+2)
生成されたSk(n+3)
Sk(n+2)の一方向圧縮を行ないWk2を得る。
Sk(n+3)の一方向圧縮を行ないWk3を得る。
Passnet更新要求メッセージ (Wk3を使う)
Passnet更新要求
不正Keyの認証メッセージ
Wk3一致?
No
Yes
Error処理
通信終了
Wk2:(Passnet), Ck1: 更新処理
生成されたSk(n+1)
Sk(n+1)の一方向圧縮を行ないWk1を得る。
秘密鍵DB
Passnet更新終了メッセージ (Wk1を使う)
Wk1一致?
Serverの Wk2(Passnet), Ck1 に更新する
One-Time Stream Cipher
Wk1(Passnet)更新シーケンス
Userの Wk2:(Passnet) Ck1: に更新する
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29

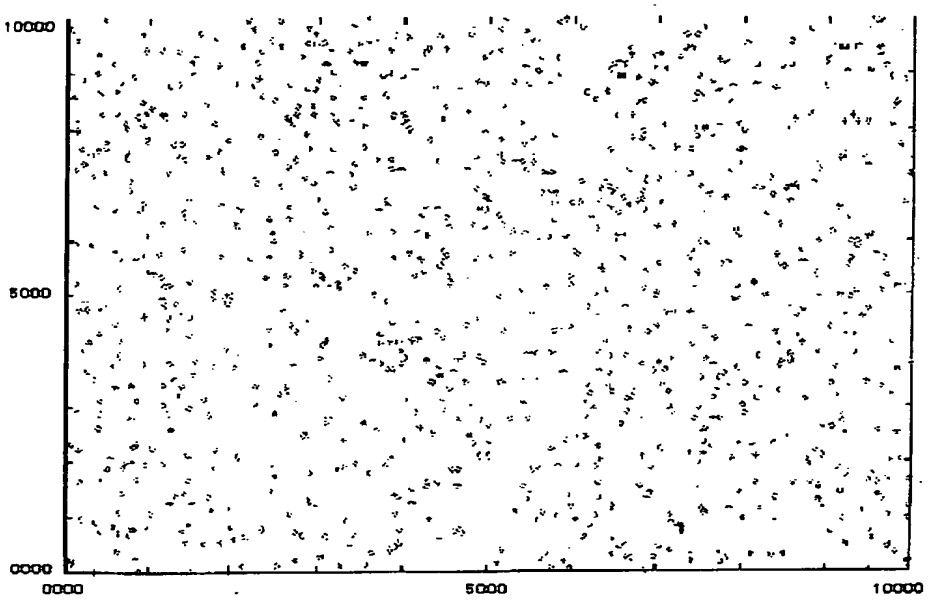
【図８】


<検証－１>
<CLIENT>
<SERVER>
登録 (User Rcgisration)
Off-line delivery／http-download
Passnet-1：ワーク鍵
Ck0：共通秘密鍵
初期認証
代理鍵 (平文)
暗号-1
鍵の同期配送
代理鍵 (平文)
Passnet-2／3 生成
交換
Passnet-2／3 生成
<サーバーを同期認証する>
<クライアントを同期認証する>
Authenticated key agreement
拡張秘密鍵を同期更新する
Passnet を同期更新する
拡張秘密鍵を同期更新する
Passnet を同期更新する
3.5-inch disk Passnet-1
Ck0：共通秘密鍵
新しい秘密鍵
新しい Passnet

【図９】


<検証－２>
<CLIENT>
<SERVER>
登録 (User Regisration)
Passnet-1
初期認証
代理鍵 (生成する)
暗号-1
Passnet-2／3 生成
<クライアントを認証する>
暗号-2
平文-2
暗号-3
平文-2
Authenticated key agreement
Authenticated key agreement
Authenticated key agreement
Authenticated key agreement
<平文-2>
<サーバーを認証する>
Passnet-2／3 生成
交換
鍵の同期配送
暗号-1
代理鍵 (平文-1)
Passnet-1
Passnet
3.5-inch disk
新しい共通鍵
新しい Passnet

【図１１】



フロントページの続き

(54)【発明の名称】　完全守秘性暗号系を構成する鍵生成システム、認証付き鍵共有プロトコル、“Ｏｎｅ－Ｔｉｍｅ　Ｓｔｒｅａｍ　Ｃｉｐｈｅｒ”、“Ｏｎｅ－Ｔｉｍｅ　ｐａｓｓｗｏｒｅｄ”及び鍵管理アルゴリズム